DS3500

Advanced Programming with Data

Northeastern University

John Rachlin
Northeastern University

Principles of Chess

The Top 35 Chess Principles

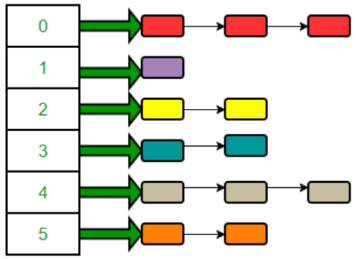
Control the Center	Doubled rooks on an open file are very strong
Develop pieces quickly	Bishops are better in open positions, knights are better in closed positions
Knights before bishops	The best way to deal with a flank attack, is with a counter attack in the center
Don't move the same piece twice in the opening	When 2 pawns can capture the same piece, capture towards the center
Don't bring your queen out too early	The king should be activated in the endgame
Castle before move 10	Rooks go behind passed pawns
Connect your rooks	2 connected passed pawns on the 6th rank will beat a rook
Rooks should go to open or half open files	Attack the base of a pawn chain
Knights on the rim are grim	Knights are usually the best piece to use to blockade a pawn
Try to avoid doubled pawns	If your position is cramped, trading pieces can help
Try to avoid isolated pawns	Trade your passive pieces for your opponent's active pieces
Try to avoid backward pawns	When ahead material, trade pieces, not pawns
Don't trade a bishop for a knight without a good reason	When behind material, trade pawns, not pieces
Avoid moving pawns in front of your castled king	Games with opposite colored bishops are dangerous in the middlegame, and drawish in the endgame
Don't open the center if your king is still there	Don't play "hope" chess
2 minor pieces are usually better than a rook and a pawn	When you see a good move, look for a better move
3 minor pieces are usually better than a queen	A really good chess player knows the right time to ignore a chess principle
Rooks are good on the 7th rank	

DS 2000: Programming with Data

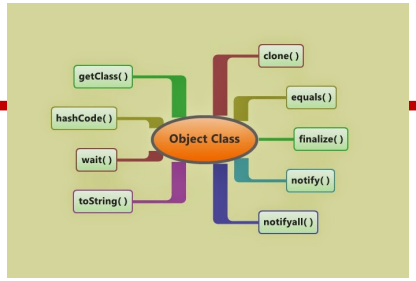
- *DS 2000* is the first course in Northeastern's *Programming with Data* sequence.
- DS 2000 is for DS majors, minors, and non-CS / non-DS majors. *No programming experience is assumed!*
- *DS 2000 is the one course to take if you are only going to take one programming course, or you want a broad introduction to Data Science.*

Some of the topics explored in DS2000:

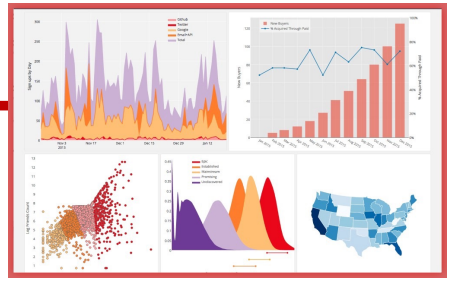
Reading and Processing Data



Data Structures and Algorithmic Thinking



Introduction to Object-Oriented Design

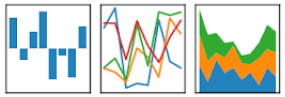


Creating Insightful Visualizations

python™

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

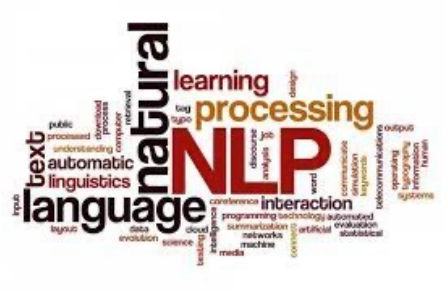


Northeastern University
Khoury College of Computer Sciences

DS 2500: Intermediate Programming with Data

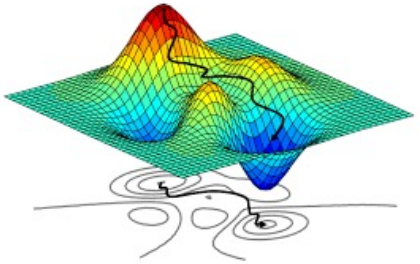
- *DS 2500* is the second course in Northeastern's *Programming with Data* sequence.
- DS 2500 is for DS majors, minors, and non-CS / non-DS majors who have taken DS 2000 or have equivalent programming experience.
- DS2500 is required for those wishing to eventually pursue more advanced data science courses such as DS3000 or DS3500.
- *DS 2500 is the course to take if you want to explore a broad range of data science topics using python.*

Some of the topics usually explored in DS2500:

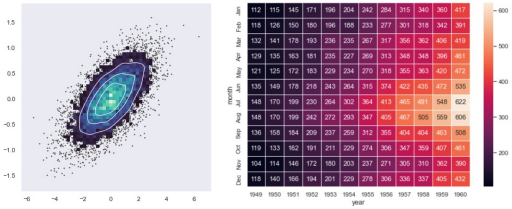


Natural Language Processing (NLP)

Graphs and Networks: Data structures and Analysis



Optimization and Machine Learning



Visualization Techniques

python™

pandas

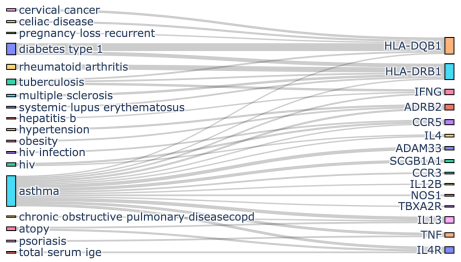
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

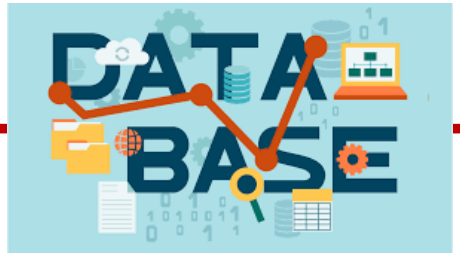
Northeastern University
Khoury College of Computer Sciences

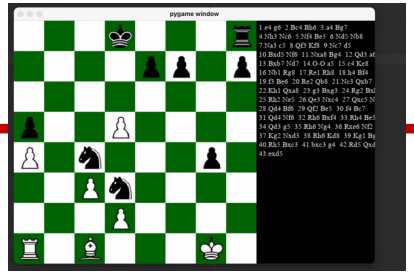
DS 3500: Advanced Programming with Data

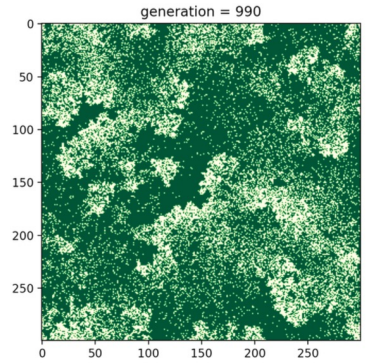
- *DS 3500* is the third and final course in Northeastern's *Programming with Data* sequence.
- *Requires DS2500: Intermediate Programming with Data*
- DS 3500 is for DS majors, minors, and non-CS / non-DS majors who have taken DS 2500 or have equivalent programming experience.
- *DS 3500 is the course to take if you want to become a professional software developer or data scientist.*

Some of the topics we will explore this Fall 2022:









Building Interactive Visualizations & Dashboards

Working with relational and non-relational databases

Object-oriented and functional paradigms.

Animation, Simulation, and Modeling

python™

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Northeastern University
Khoury College of Computer Sciences

DS3500 Topics

Tooling

- Managing environments
- Anaconda library management
- The *git* software control system
- The JetBrains PyCharm IDE

Techniques

- Exception handling
- Object-oriented design patterns
- Test-Driven Development
- Functional programming
- Decorator patterns / Profiling
- Type hints
- Extensibility

Modules

- Advanced visualization with plot.ly
- Modeling and simulation / Animation
- Risk analysis for business forecasting
- Building interactive dashboards
- Databases / Persistence
- Optimization and Intelligent Decision Support
- Evolutionary Computing
- Scientific computing / multi-core computing

- Working with streaming data
- Time-series analysis
- Data structures: Property graphs for Network Analysis
- Image processing
- Concurrency with *asyncio*
- Web-development / User-interfaces with *flask*
- Other?

DS3500 –Programming with Data

Setting up your Python development environment

Northeastern University

Three critical elements

Anaconda: A python distribution and environment manager (or "package manager") that comes pre-installed with a *base* environment containing hundreds of packages. (Mine currently has 278!)

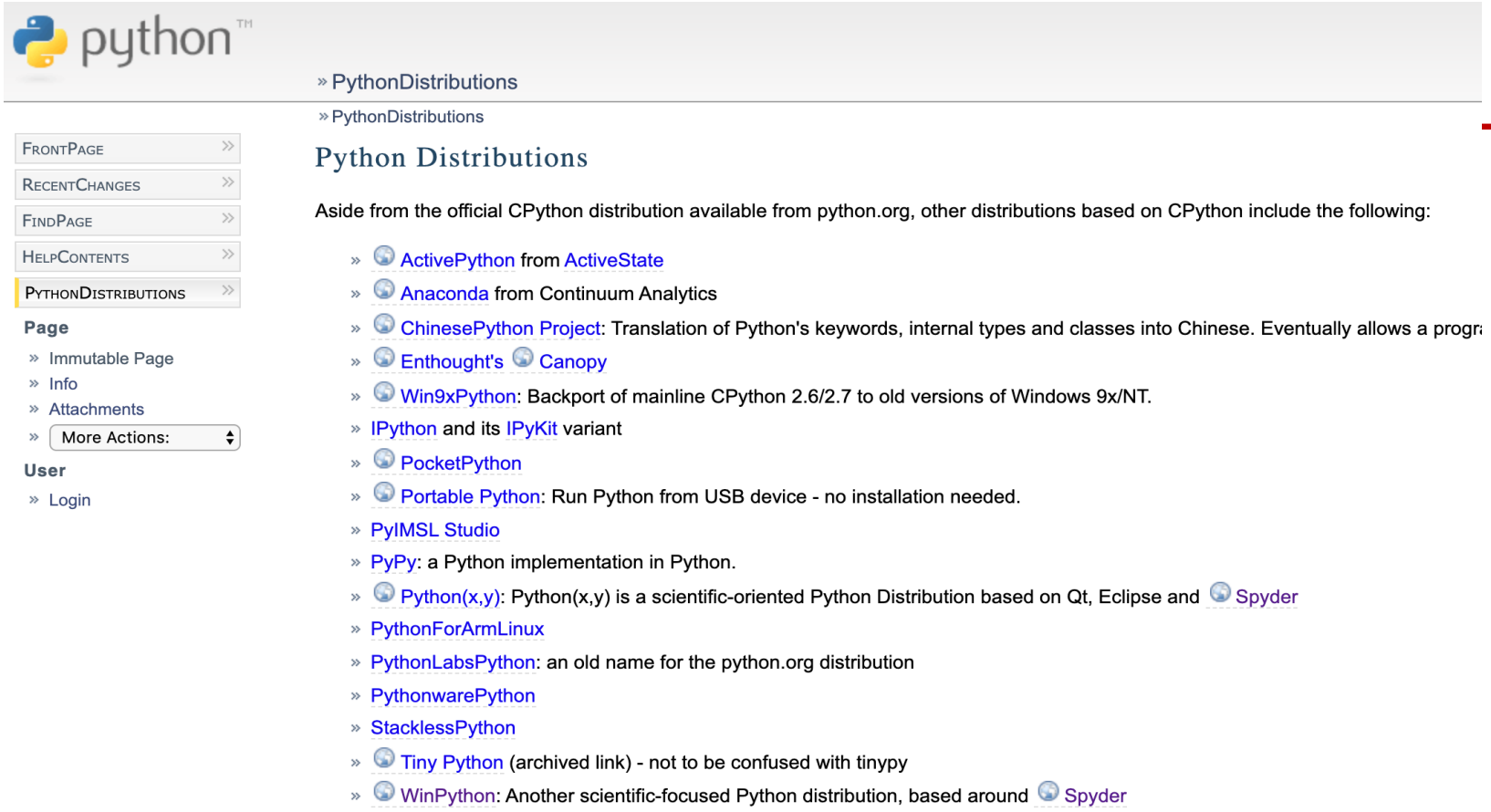
An **IDE** (Integrated Development Environment): For creating projects and writing, running, and testing code. We'll be mostly using **Jetbrains PyCharm** in class but I might sometimes revert to spyder or Jupyter Notebooks. PyCharm readily integrates with Anaconda environments.

GIT: A software control system – for working together on shared development projects. PyCharm integrates with GIT.

Python Distributions

A python *distribution* typically consists of:

- The python programming language
- Development tools
- Pre-installed python packages (libraries) that make python more powerful
- A package manager (allows you to customize your distribution by adding more libraries or updating existing libraries)



python™

» PythonDistributions

» PythonDistributions

FrontPage
RecentChanges
FindPage
HelpContents
PythonDistributions

Page

» Immutable Page
» Info
» Attachments
» More Actions:

User

» Login

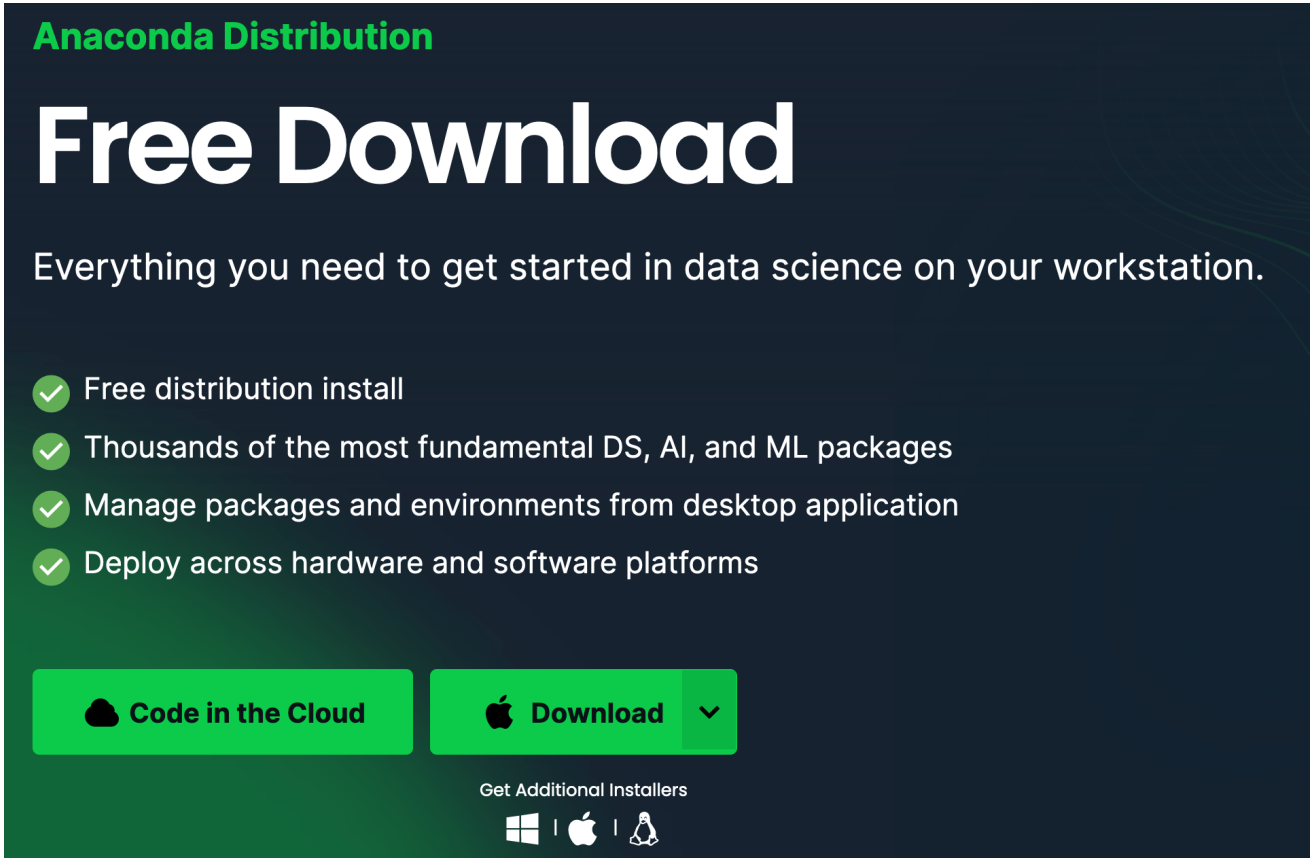
Python Distributions

Aside from the official CPython distribution available from python.org, other distributions based on CPython include the following:

» ActivePython from ActiveState
» Anaconda from Continuum Analytics
» ChinesePython Project: Translation of Python's keywords, internal types and classes into Chinese. Eventually allows a progr
» Enthought's Canopy
» Win9xPython: Backport of mainline CPython 2.6/2.7 to old versions of Windows 9x/NT.
» IPython and its IPyKit variant
» PocketPython
» Portable Python: Run Python from USB device - no installation needed.
» PyIMSL Studio
» PyPy: a Python implementation in Python.
» Python(x,y): Python(x,y) is a scientific-oriented Python Distribution based on Qt, Eclipse and Spyder
» PythonForArmLinux
» PythonLabsPython: an old name for the python.org distribution
» PythonwarePython
» StacklessPython
» Tiny Python (archived link) - not to be confused with tinypy
» WinPython: Another scientific-focused Python distribution, based around Spyder

There are other python distributions out there, but we'll use Anaconda.

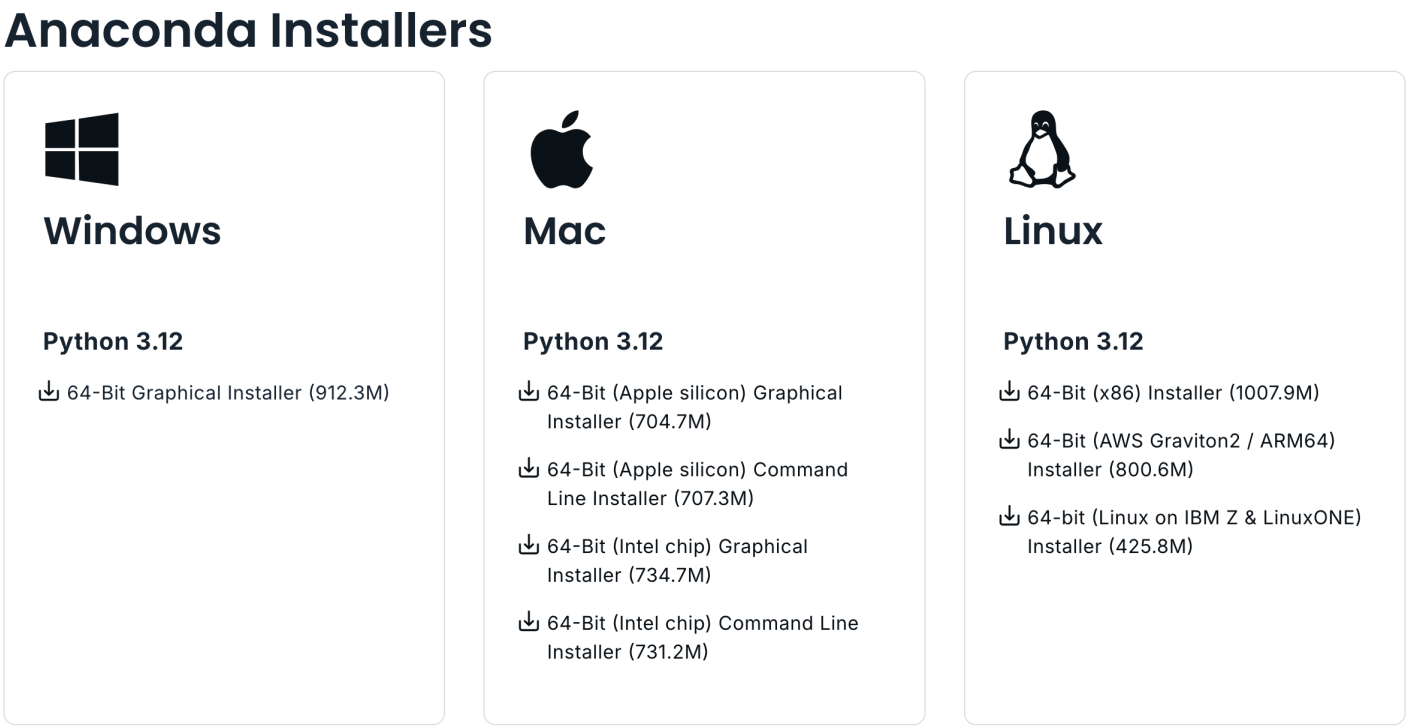
www.anaconda.com/download



Anaconda installers

https://www.anaconda.com/download#downloads

(Skip Registration if you want.)



Installing Anaconda on a Mac

Anaconda3-2023.07-2-MacOSX-x86_64.pkg

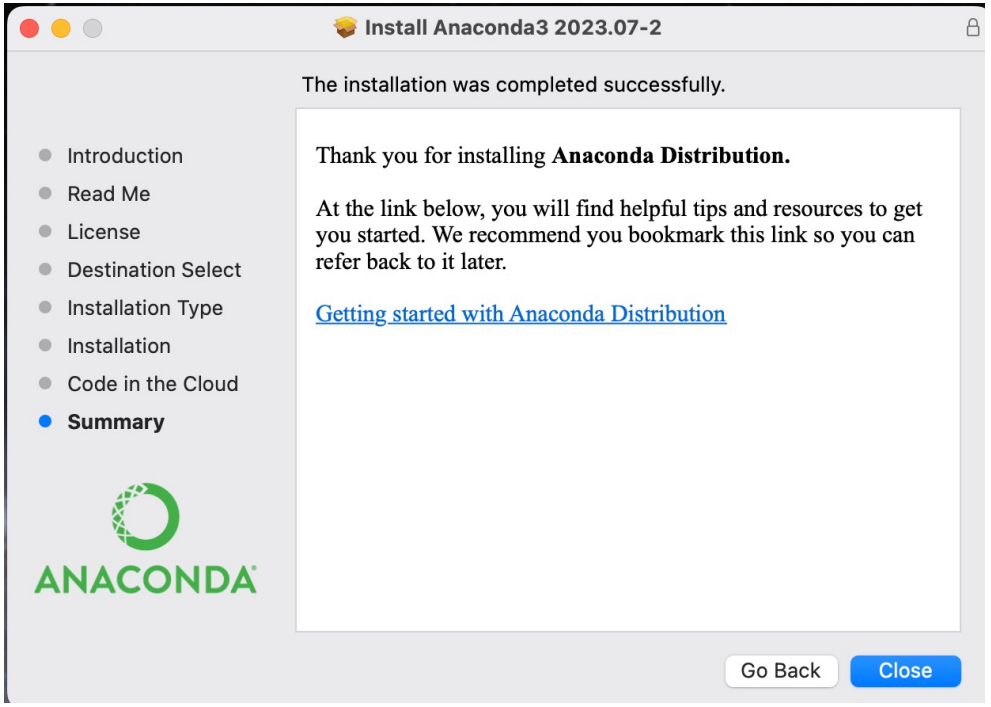
Completed — 611 MB

I chose /Users/rachlin/opt as my installation folder. The home directory for anaconda is then:

/Users/rachlin/opt/anaconda3

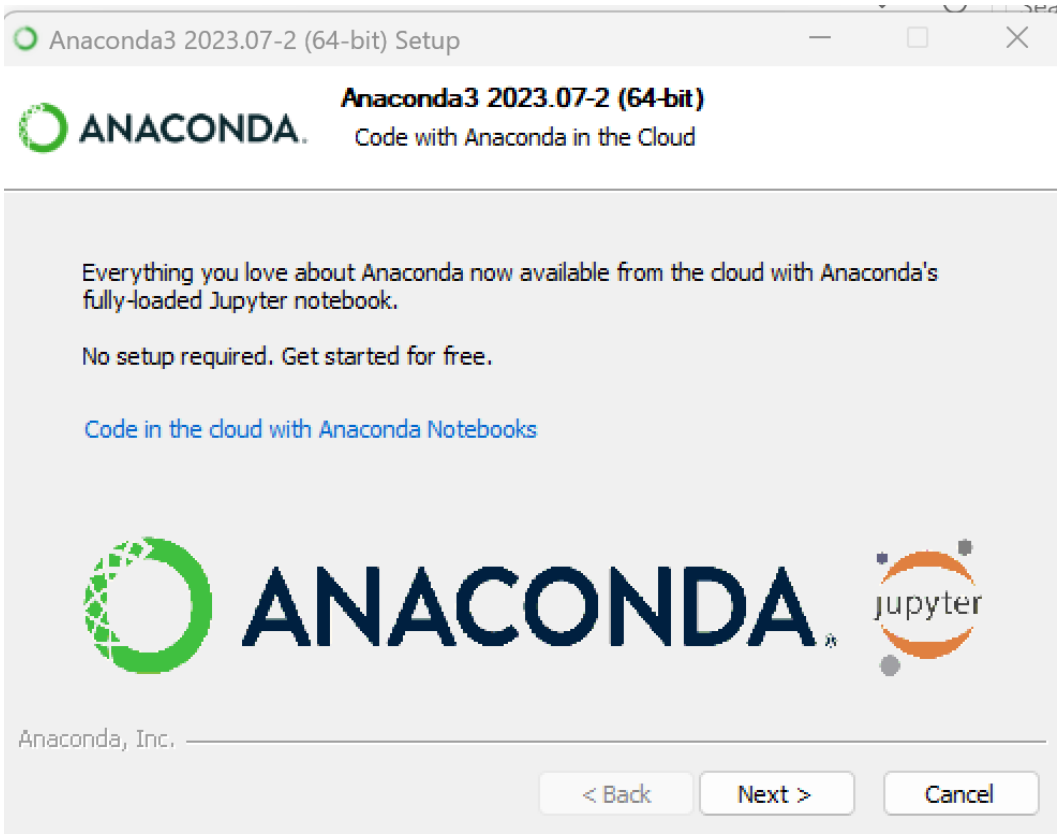
It installs Anaconda Navigator and tries to get you to set up a cloud account. I don't use it.

The "base" environment uses python 3.11.4

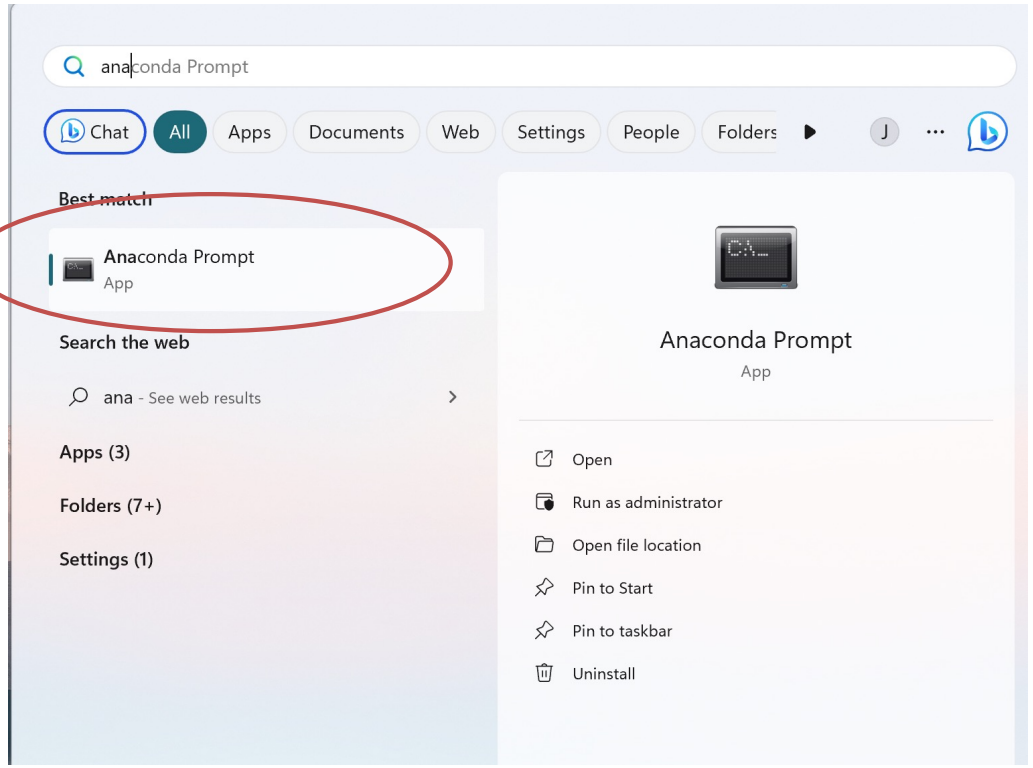


Anaconda on Windows

I just accepted defaults or any recommended buttons when installing on windows.



Running conda commands on Windows



To run “conda” commands on windows, use the Anaconda Prompt. (You might want to pin this application to your Task Bar.)

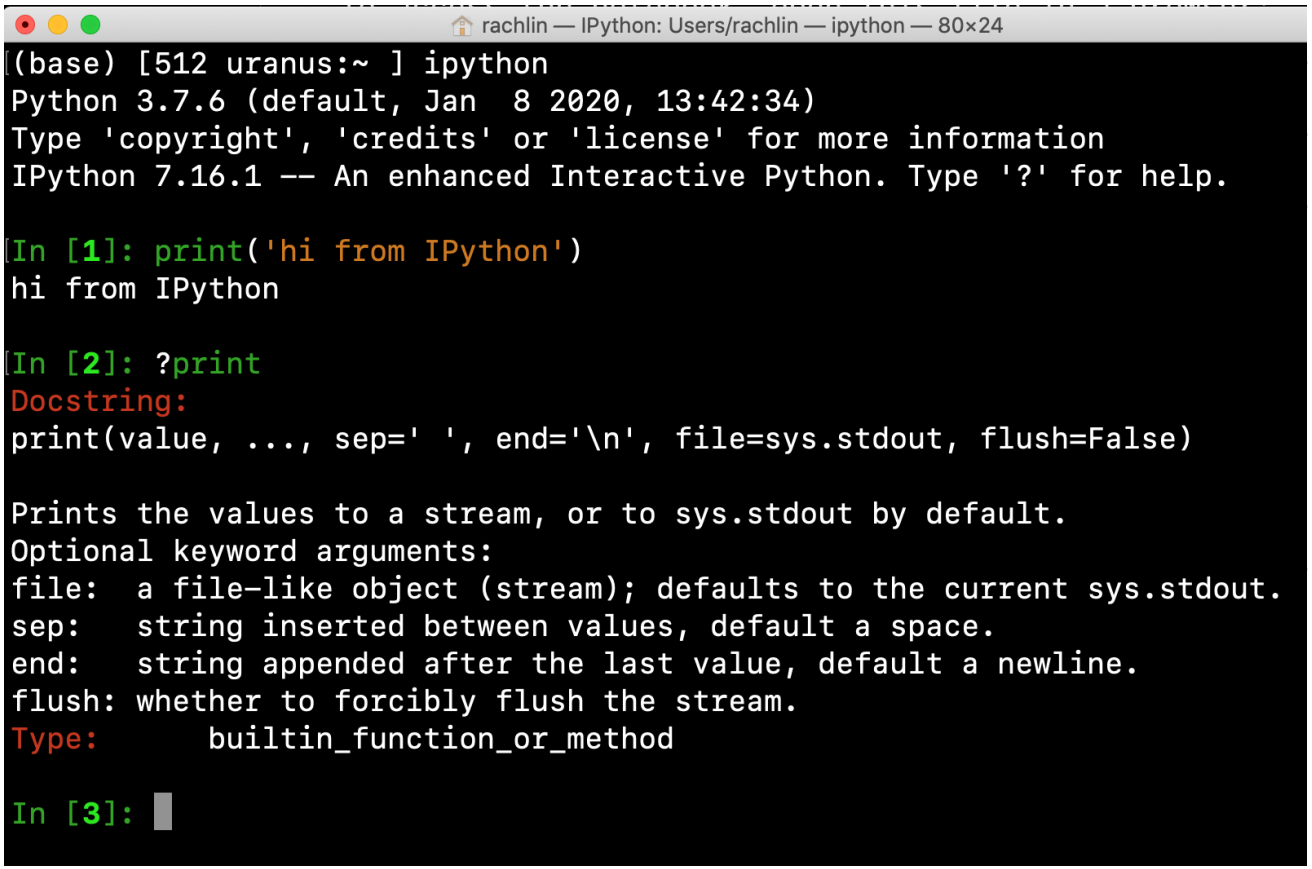
It basically opens a command terminal with the base anaconda environment activated.

conda info

Note that the base environment lives in the anaconda home directory. Other environments live in an **envs** subdirectory.

```
(base) [501 uranus:~ ] conda info

     active environment : base
    active env location : /Users/rachlin/opt/anaconda3
            shell level : 1
       user config file : /Users/rachlin/.condarc
 populated config files : /Users/rachlin/.condarc
          conda version : 23.7.2
    conda-build version : 3.26.0
         python version : 3.11.4.final.0
       virtual packages : __archspec=1=x86_64
                          __osx=10.16=0
                          __unix=0=0
       base environment : /Users/rachlin/opt/anaconda3  (writable)
      conda av data dir : /Users/rachlin/opt/anaconda3/etc/conda
  conda av metadata url : None
           channel URLs : https://repo.anaconda.com/pkgs/main/osx-64
                          https://repo.anaconda.com/pkgs/main/noarch
                          https://repo.anaconda.com/pkgs/r/osx-64
                          https://repo.anaconda.com/pkgs/r/noarch
          package cache : /Users/rachlin/opt/anaconda3/pkgs
                          /Users/rachlin/.conda/pkgs
       envs directories : /Users/rachlin/opt/anaconda3/envs
                          /Users/rachlin/.conda/envs
                platform : osx-64
              user-agent : conda/23.7.2 requests/2.31.0 CPython/3.11.4 Darwin/22.6.0 OSX/10.16
                 UID:GID : 501:20
              netrc file : None
            offline mode : False
```

IPython: An enhanced REPL for Python



Syntax highlighting
Code completion
Documentation

You'll want to install IPython in your **ds** environment. It's useful.

Create a “ds” environment

The base environment comes with almost 500 libraries, most of which you don’t need. Run the following commands to set up a **ds** environment with a few starting python libraries.

```
conda create --name ds python=3.11
conda activate ds
conda install matplotlib
conda install pandas
conda install ipython
```

Checking that I have two environments, and where they are located

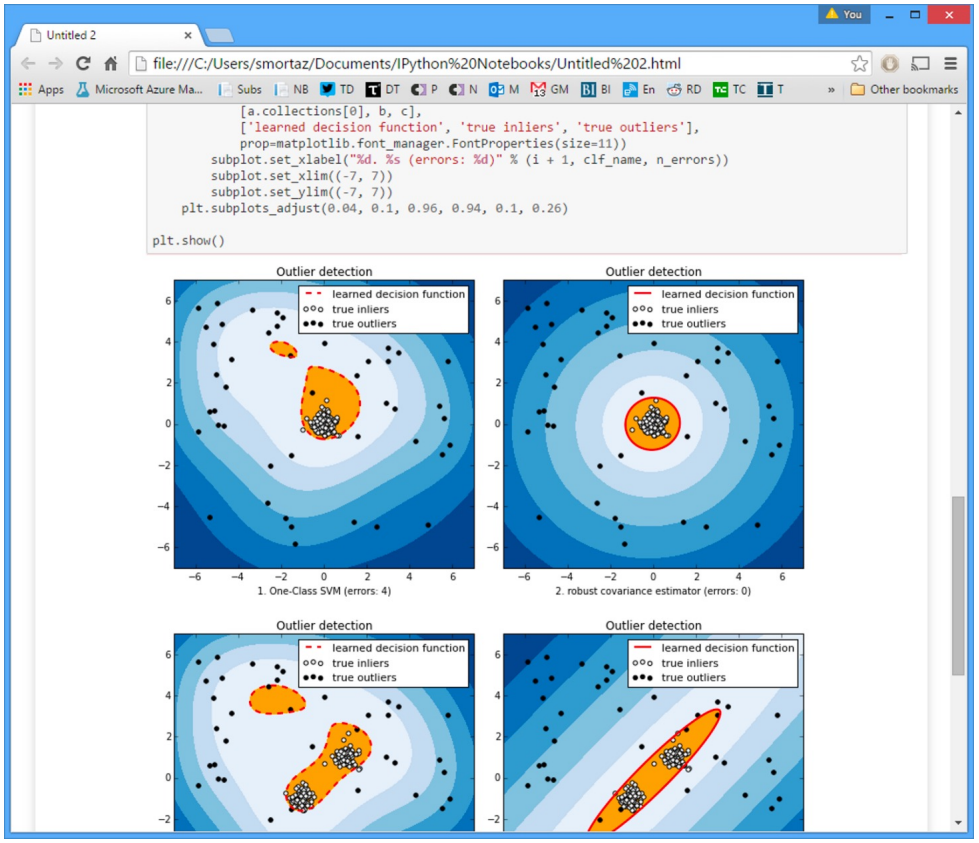
```
conda env list

# conda environments:
#
base                     /Users/rachlin/opt/anaconda3
ds                    *  /Users/rachlin/opt/anaconda3/envs/ds
```

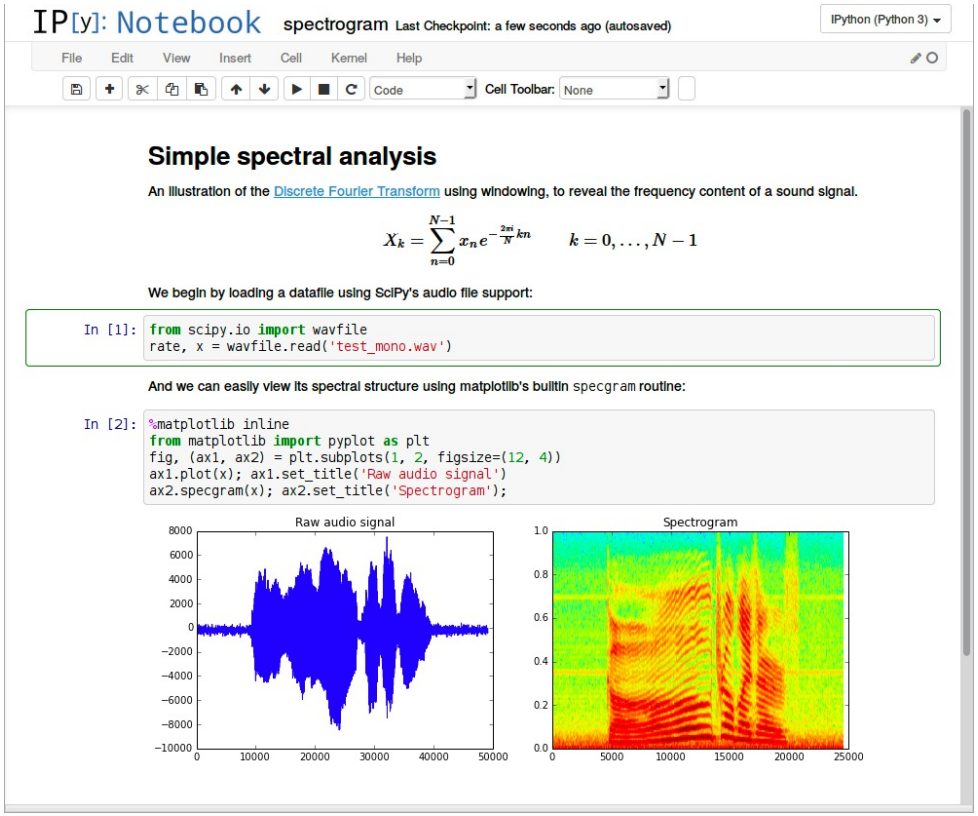
Environment locations on Windows

On windows, it is the same (but I have a different username).

```
(ds) C:\Users\johnr>conda env list
# conda environments:
#
base                     C:\Users\johnr\anaconda3
ds                    *  C:\Users\johnr\anaconda3\envs\ds
```

Jupyter Notebooks: Interactive Python for Data Science

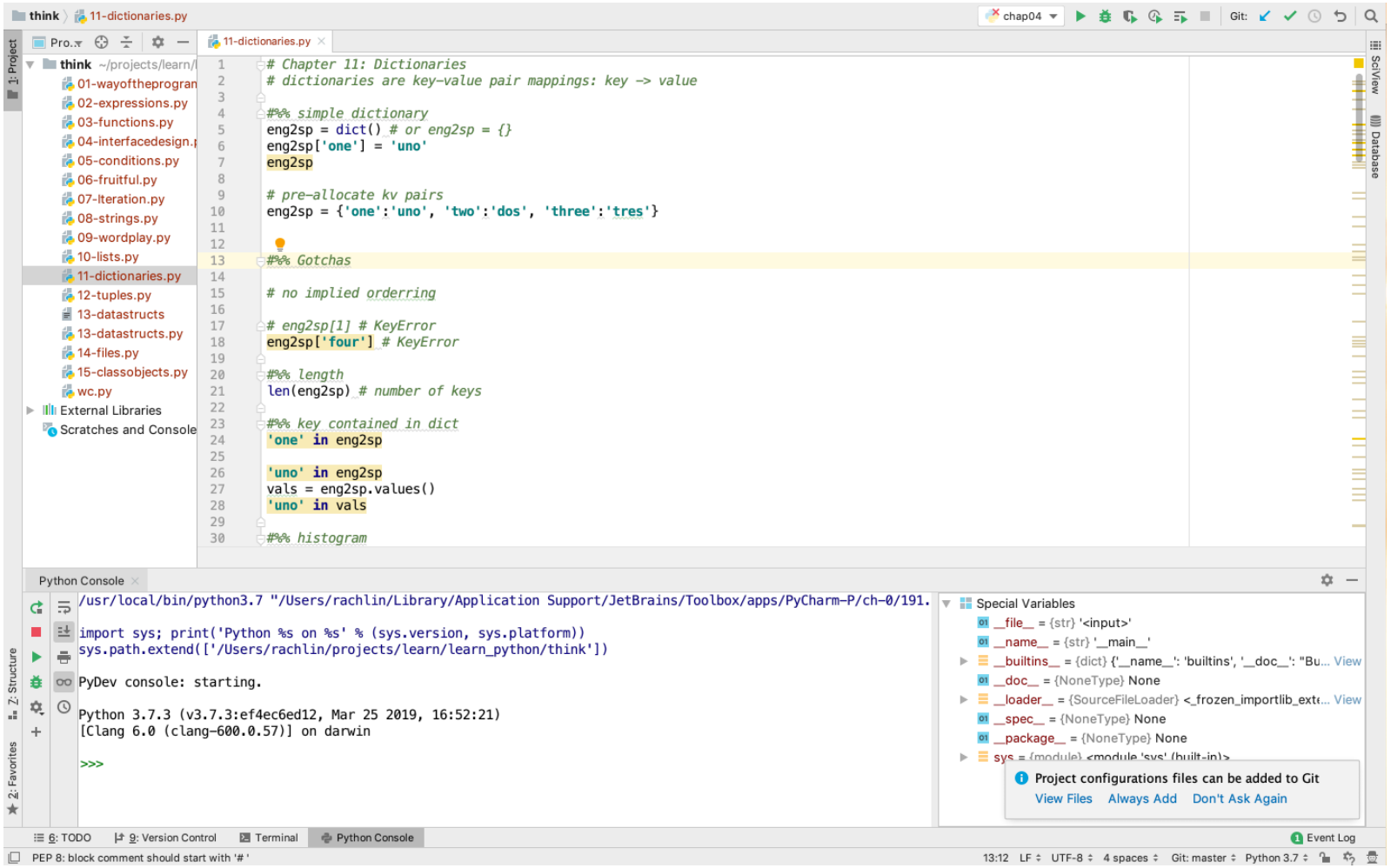




Northeastern University

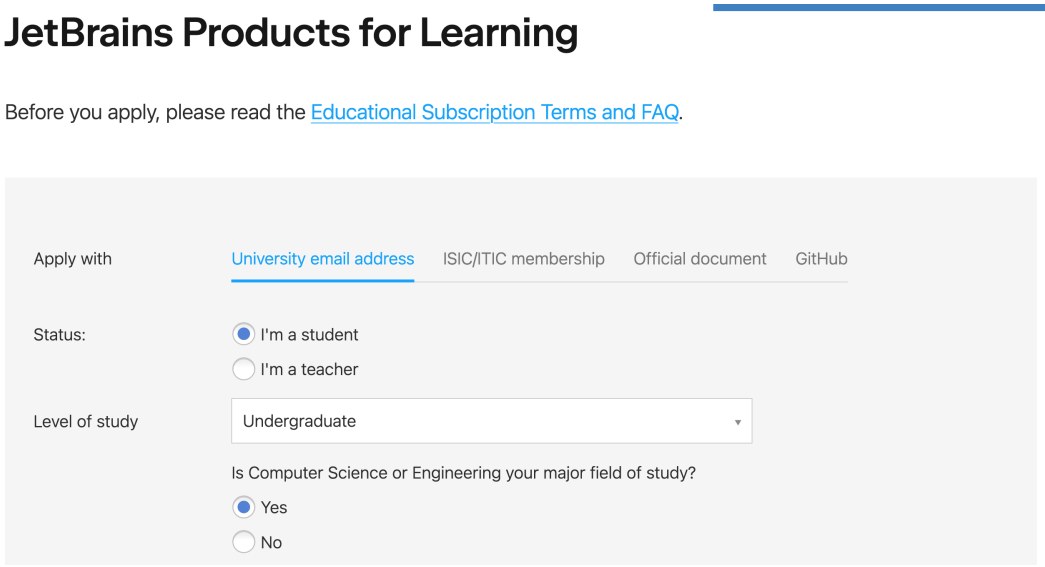
Jupyter is useful for data science but not for the types of programs we'll be writing in this class where we focus on code reuse by building *libraries.*

PyCharm (JetBrains): Recommended Python IDE



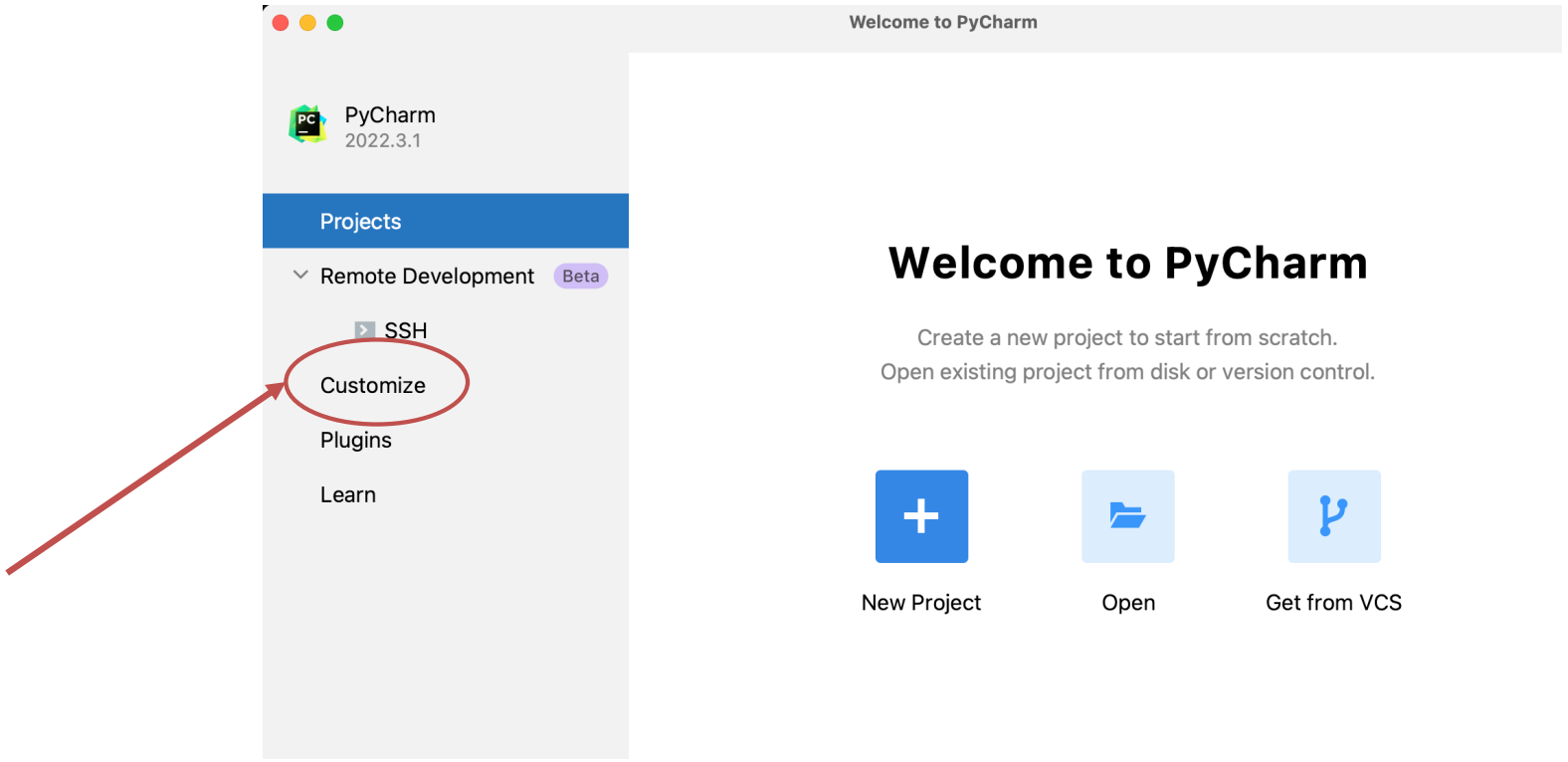
PyCharm: Community or Professional?

https://www.jetbrains.com/shop/eform/students

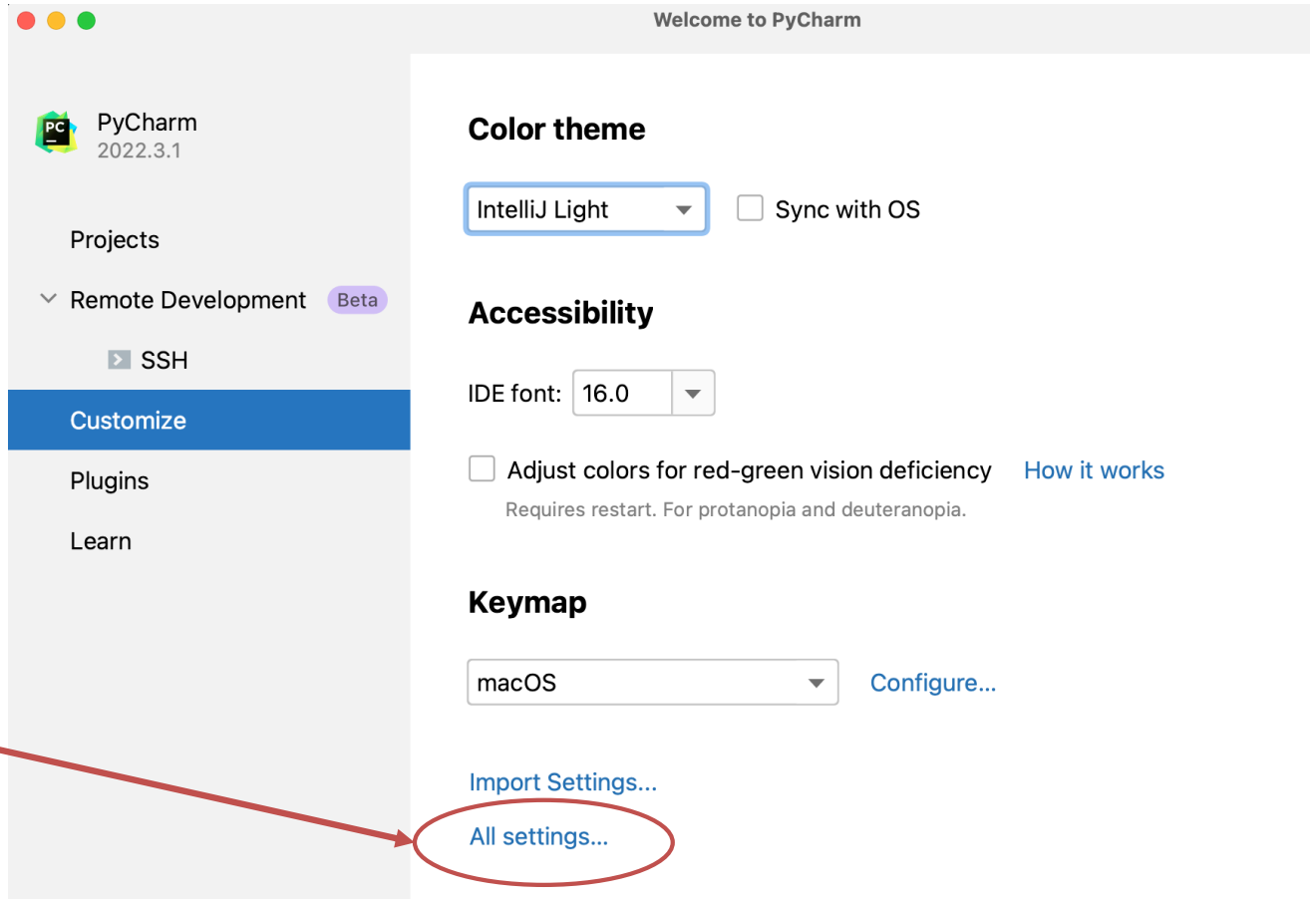


The community edition should be fine for this class, but you can get access to the professional edition and other JetBrains products for free by signing up for an Educational License (Free).

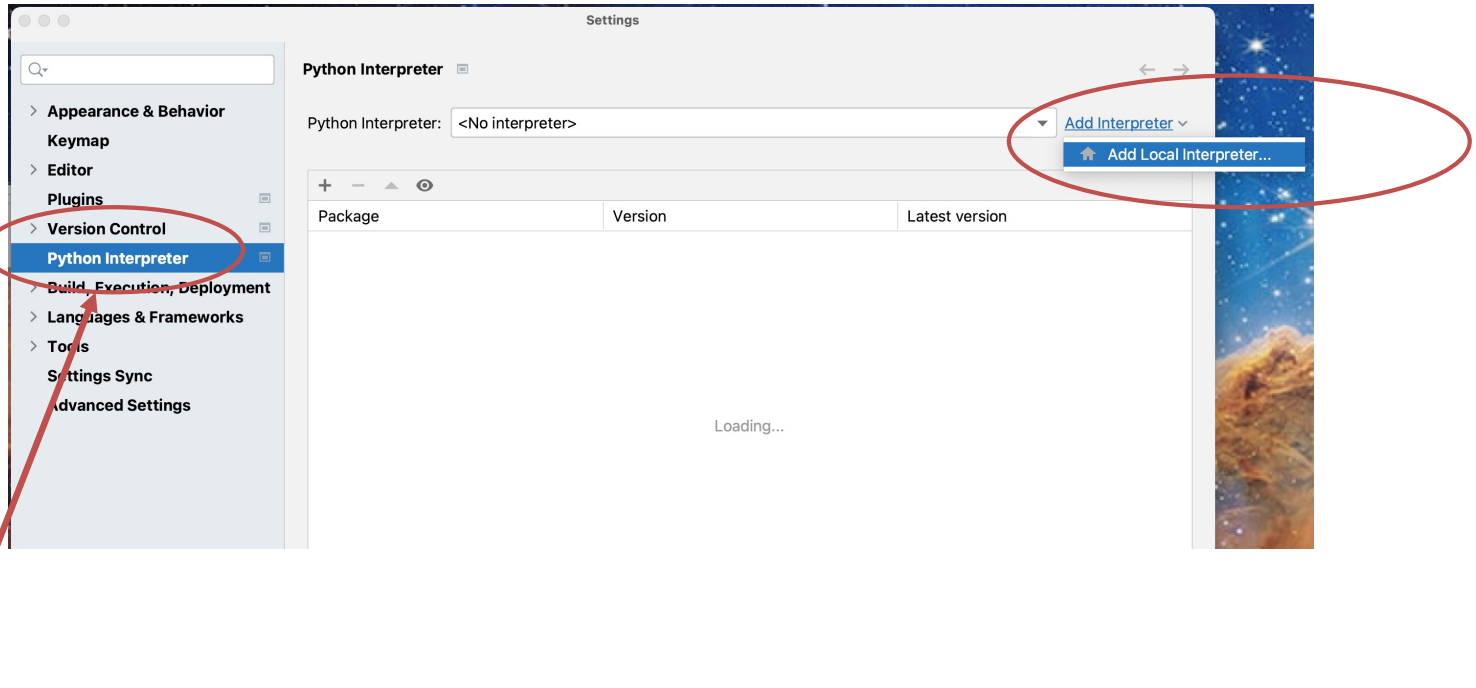
Connecting PyCharm project to an environment



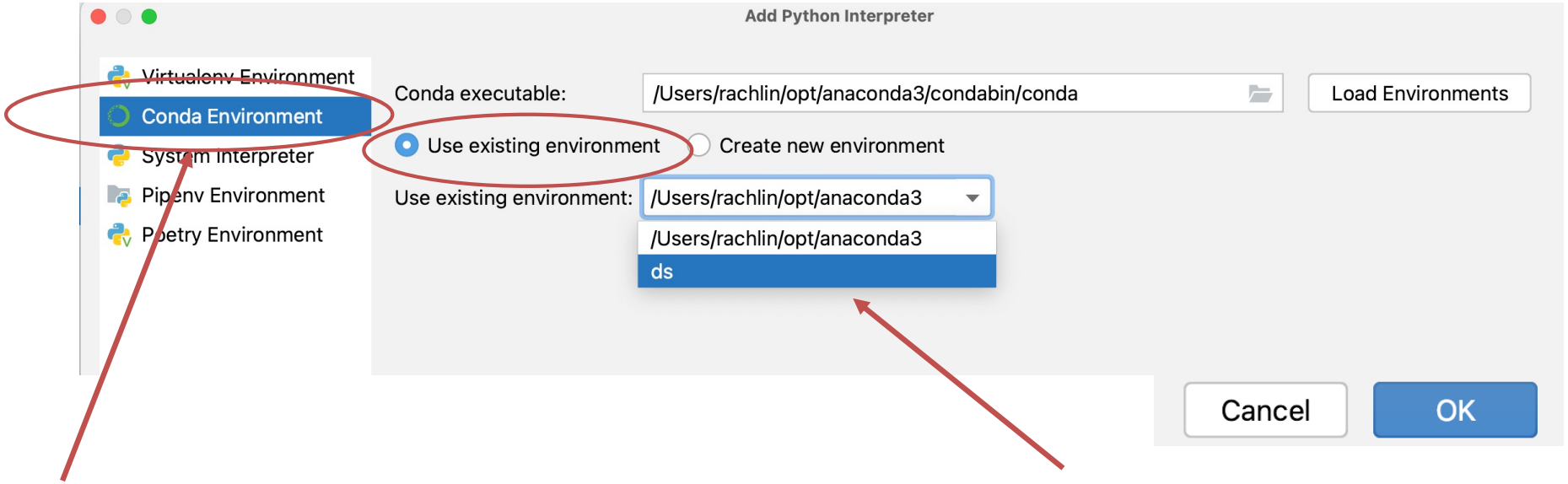
Connecting PyCharm project to an environment



Connecting PyCharm project to an environment

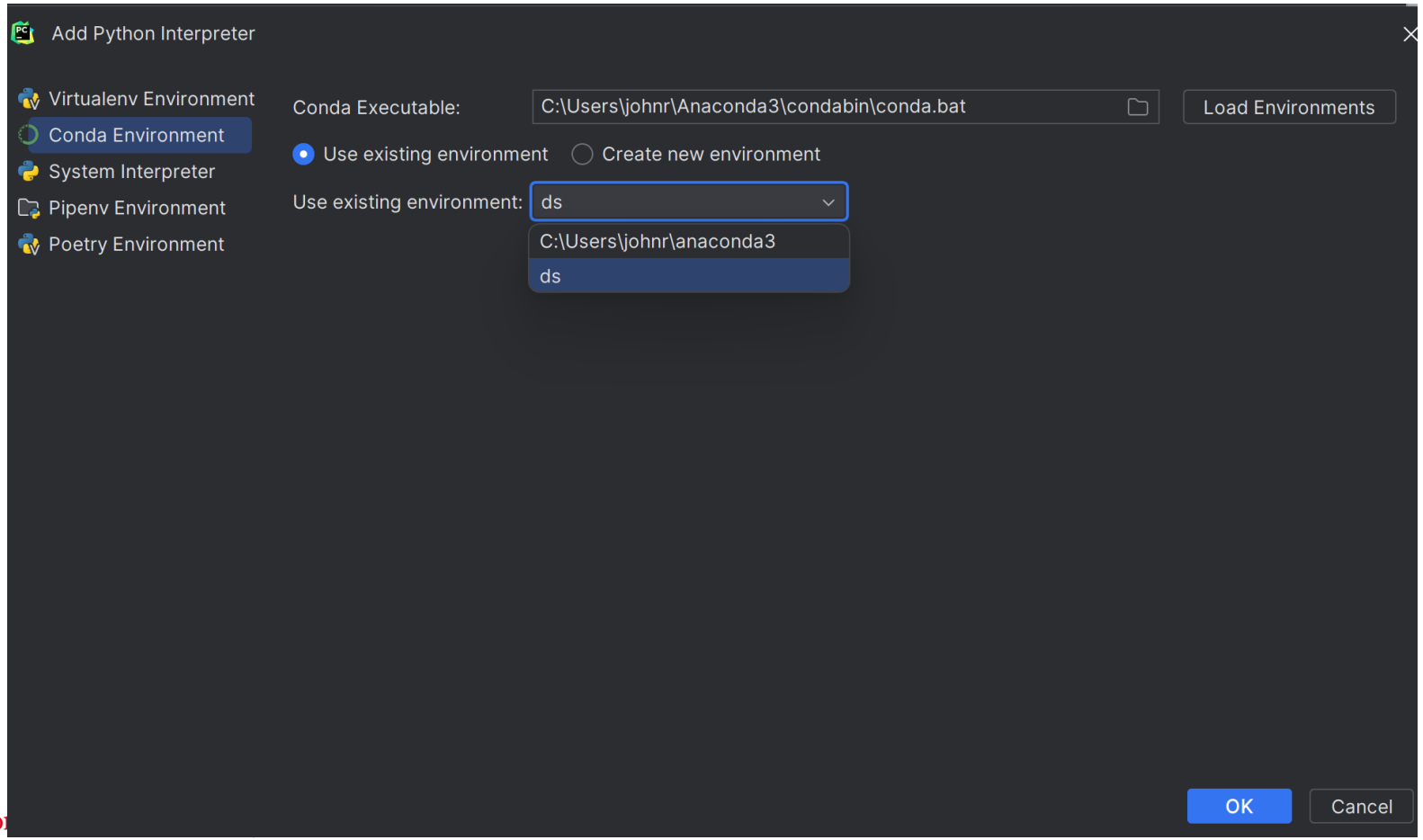


Connecting PyCharm project to an environment

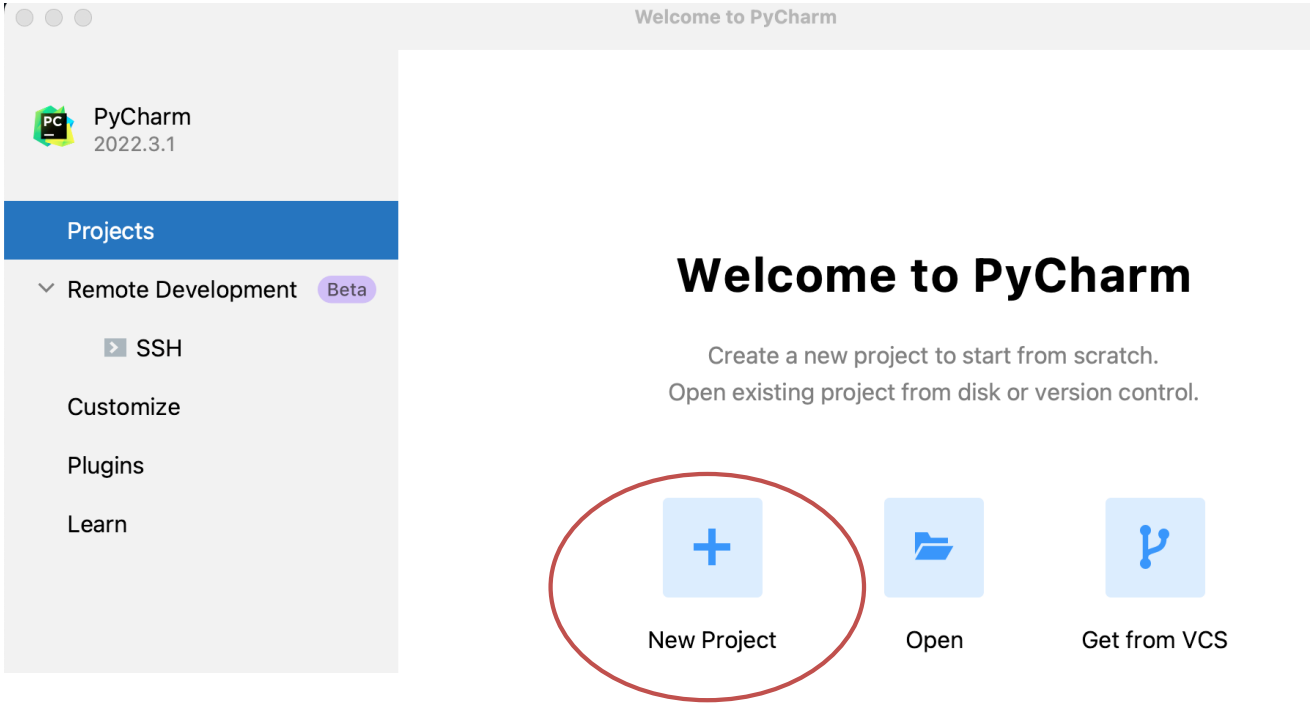


Select the “ds” environment you created and then press OK

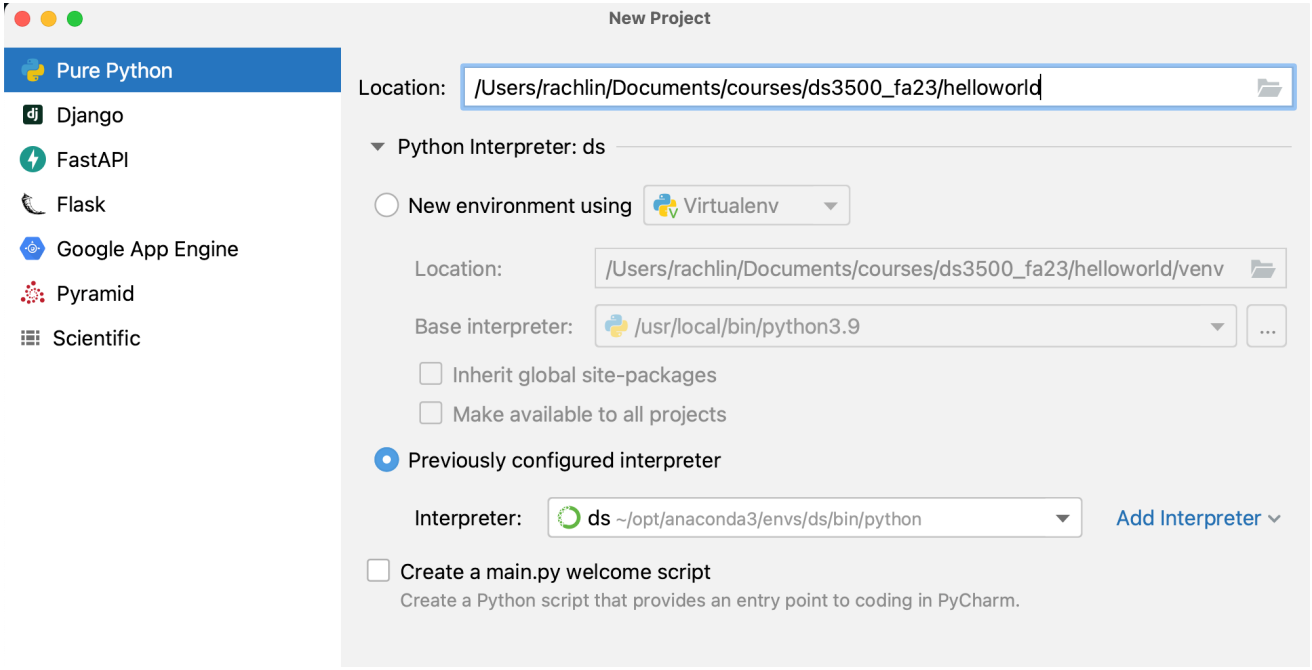
Very similar in Windows



Create your first project

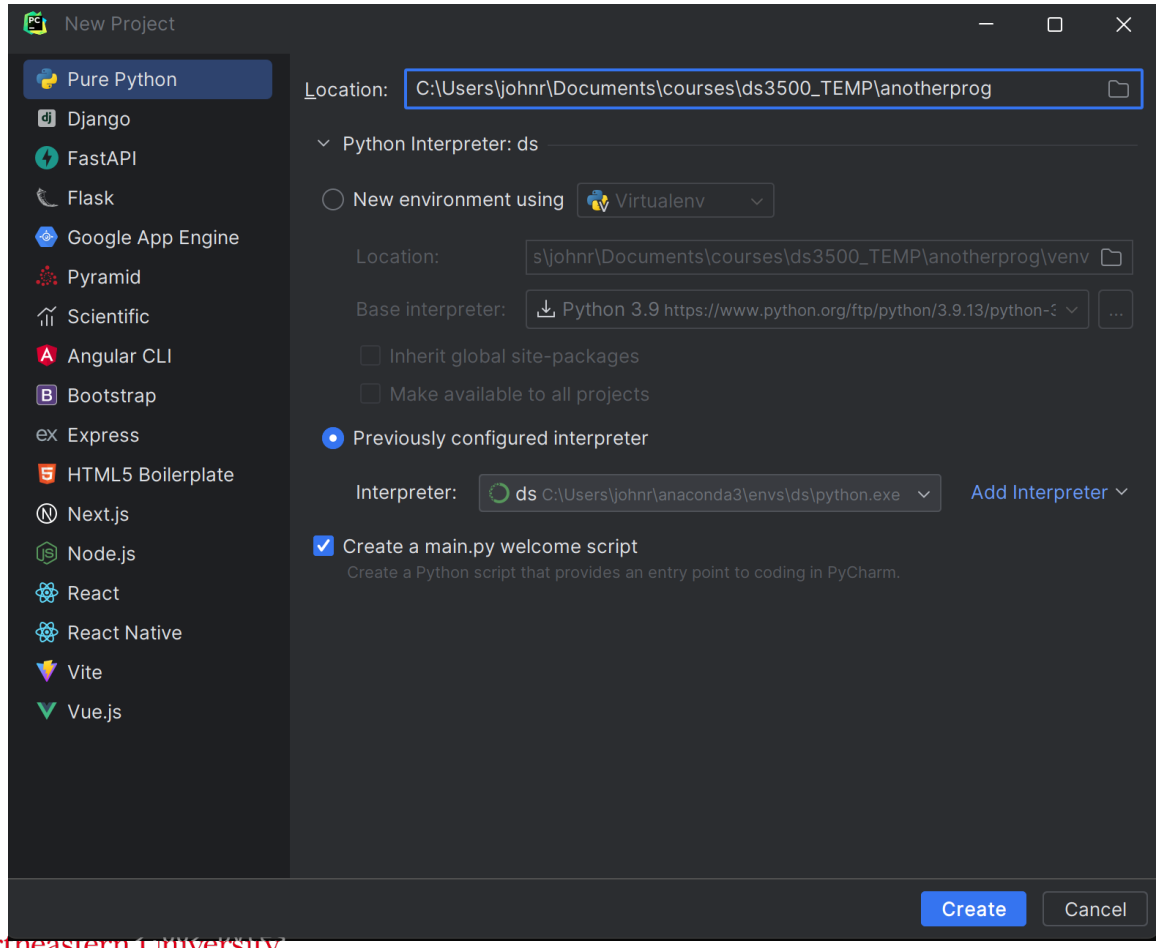


Create your first project



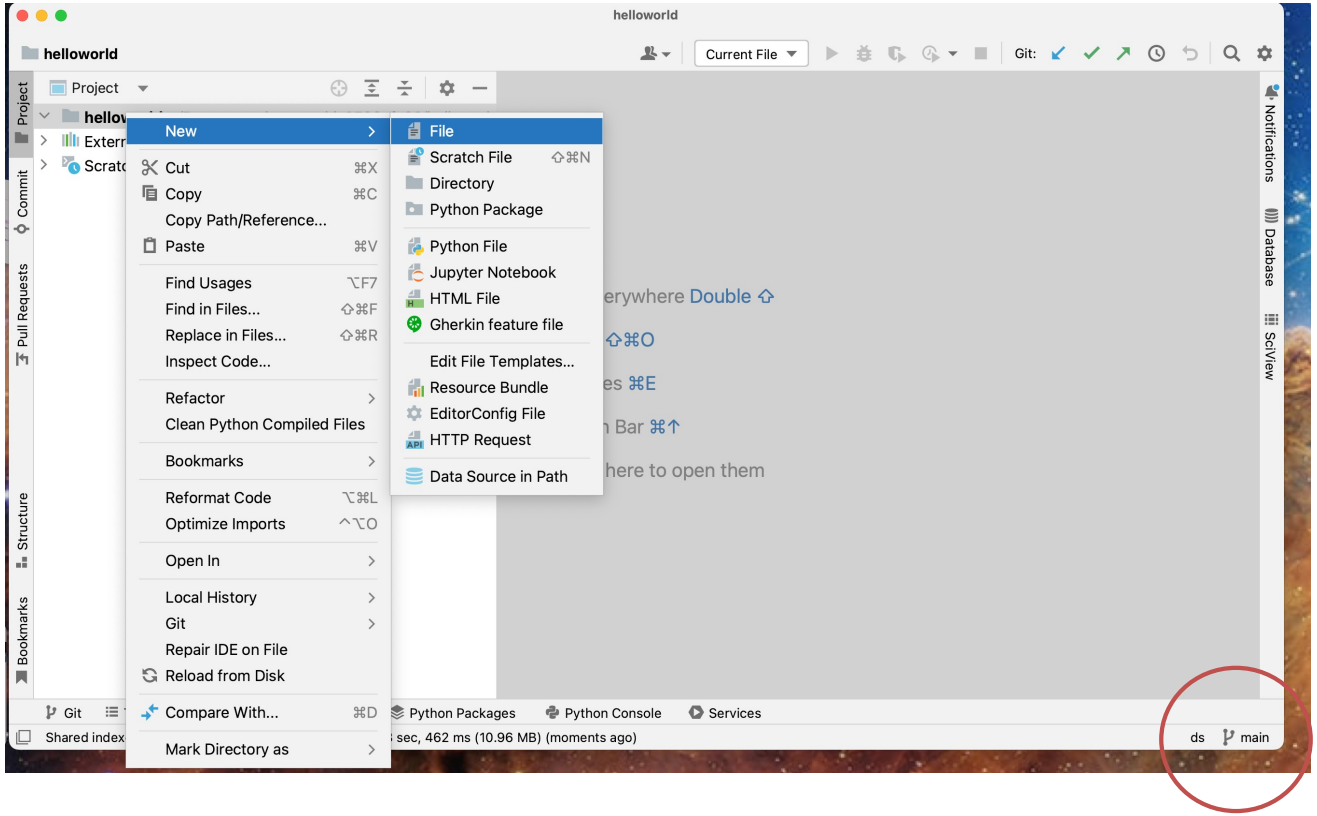
Here I'm creating a project folder in my local copy of the class repo (ds3500_fa23).
You would create YOUR projects and code outside the repo folders – perhaps in your OWN repo!

Windows: Creating a New Project (same)



I created this project
in a different repo / folder:
ds3500_TEMP

Creating our first python program



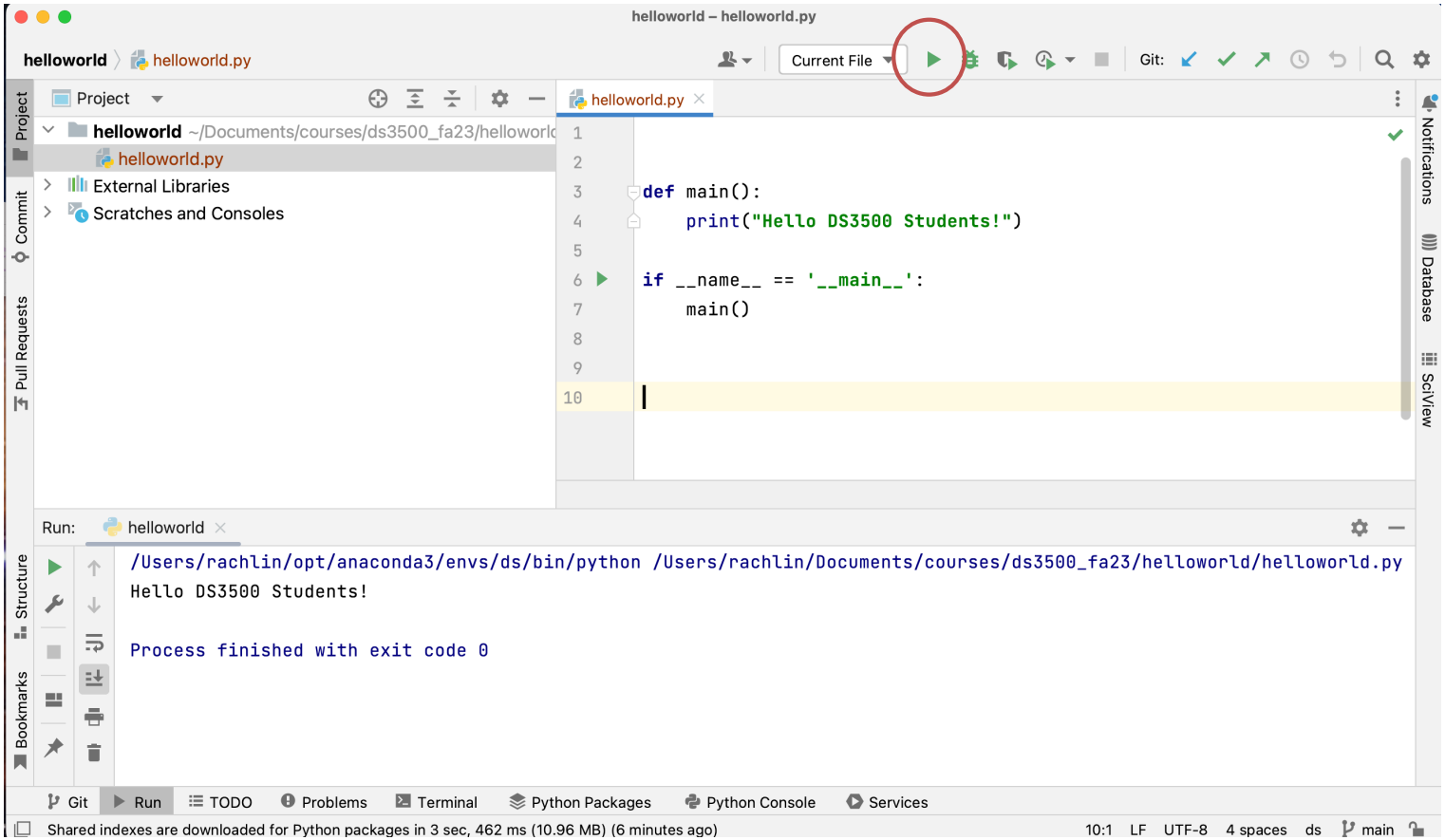
New File

helloworld.py

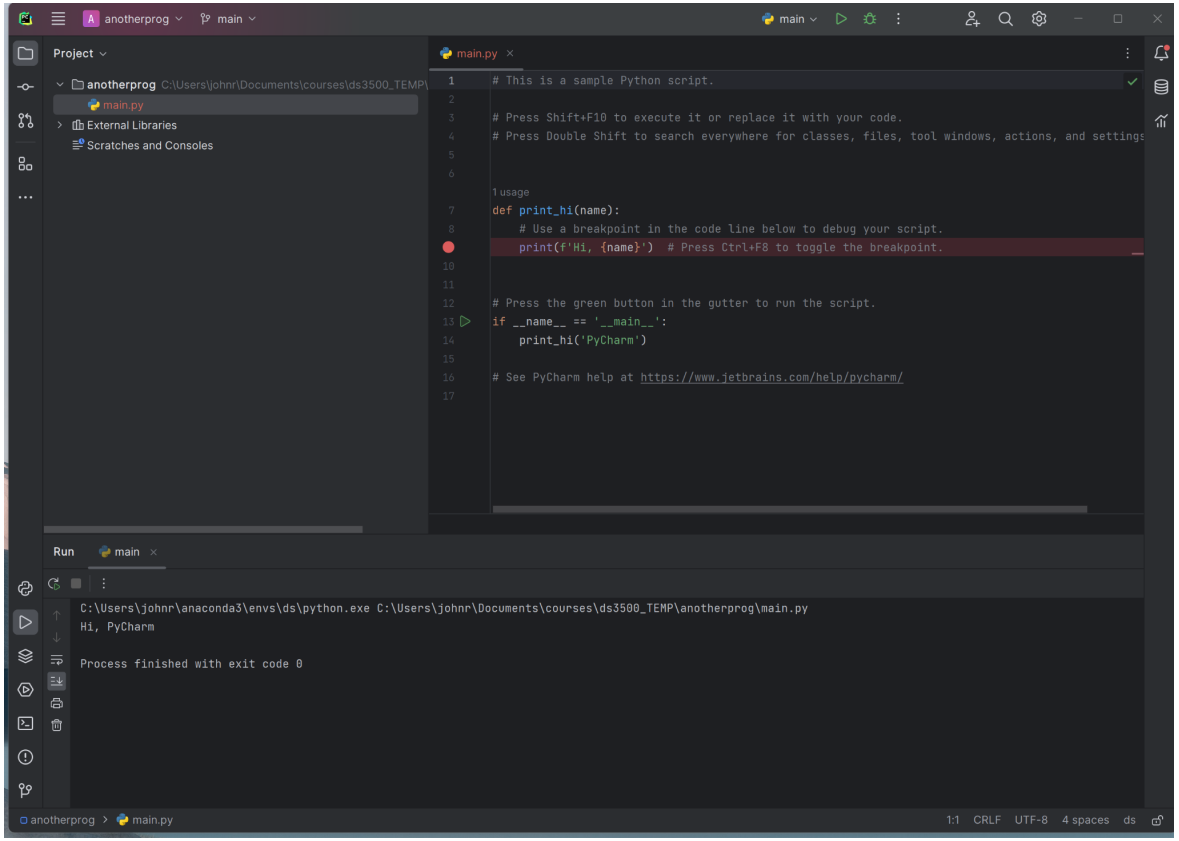
Right-clicking on the project name (**helloworld**) and adding a new file which will be called *helloworld.py.* Notice that “ds”, our active environment is listed in the status bar.

PyCharm knows this is a git repo and it is possible to manage your repo through PyCharm. I prefer to use the command line, so I will decline when PyCharm asks me about tracking newly created files.

Running our program



Windows Interface (dark theme)



You can choose different themes and syntax highlighting schemes.

Python Development Tools - Summary

Tool / Platform	Strengths and Weaknesses
IPython (RECOMMENDED)	Great for testing small snippets of Python and for learning Python syntax. Not intended for application development. IPython is an enhanced REPL for Python that comes packaged with the Anaconda distribution.
IDLE (NOT FOR ADVANCED)	Start coding today! Comes with Python distribution. Fine for small, single-file, stand-alone scripts or applications. Not recommended for more complex projects.
Atom text editor (OBSOLETE)	An extendible text-editor with many plugins for Python programming. See Prof. Rachlin's handout on recommended plugins.
Spyder (LIMITED)	A full-fledged IDE installed with the Anaconda distributions. Manage multiple files. Monitor variables. Run isolated blocks of code.
PyCharm (JetBrains) (RECOMMENDED)	Another popular IDE for python. Has more functionality than what you will need for this class. I won't be making use of PyCharm in this class, but you are welcome to explore its capabilities and use it for your homework.
Jupyter notebooks (PROTOTYPING ONLY)	A *notebook-style* development environment popular with data scientists. Integrate code, text, tables, visualizations in one file. Not suited for software *applications* requiring multiple python files.
Anaconda (REQUIRED)	A Python distribution that comes with Python, popular pre-installed libraries, and a variety of development tools including Spyder, Ipython, and Jupyter notebooks. The Anaconda Navigator is a jumping-off point for applications, learning resources, and community forums. **Recommended.**

Installing Git on a Mac

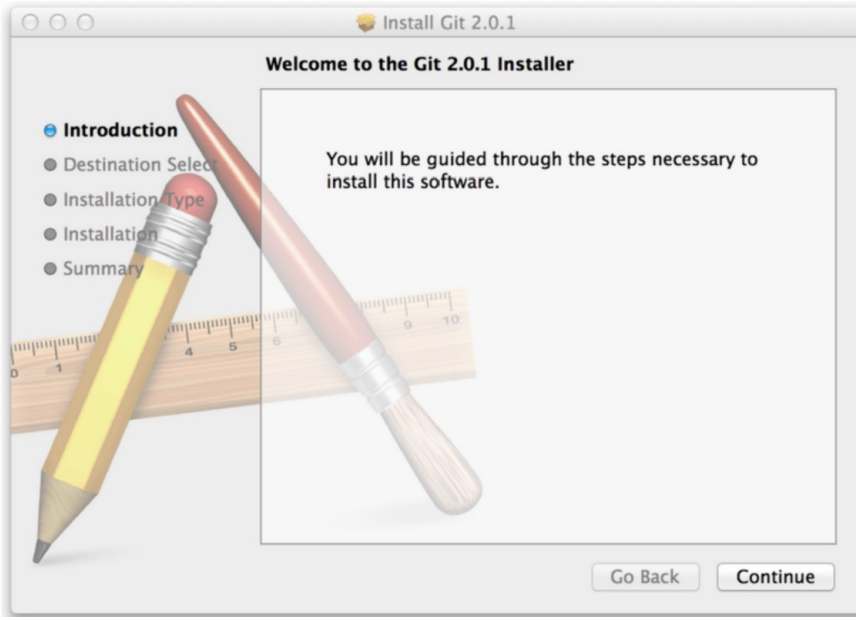
The Git Pro books has instructions: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

There are several ways to install Git on macOS. The easiest is probably to install the Xcode Command Line Tools. On Mavericks (10.9) or above you can do this simply by trying to run `git` from the Terminal the very first time.

```
$ git --version
```

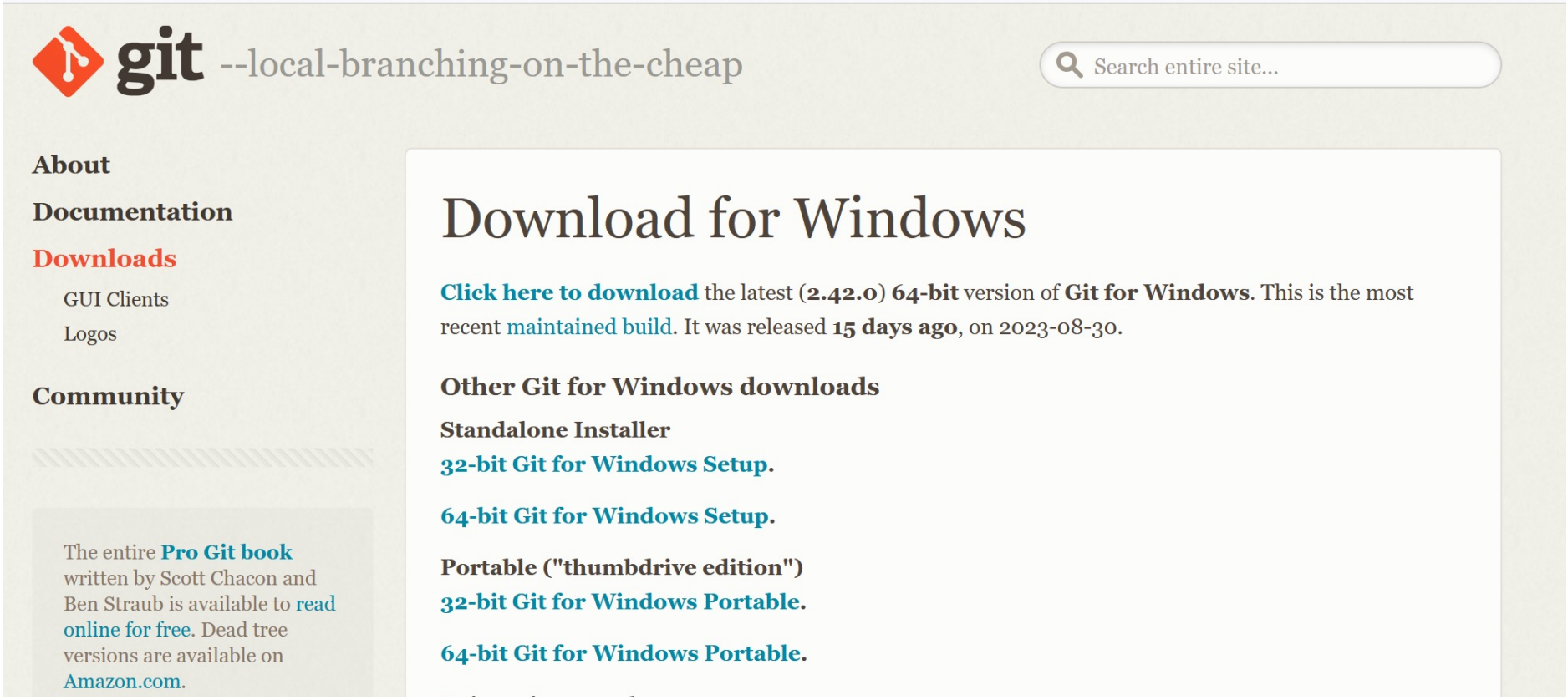
If you don't have it installed already, it will prompt you to install it.

If you want a more up to date version, you can also install it via a binary installer. A macOS Git installer is maintained and available for download at the Git website, at https://git-scm.com/download/mac.



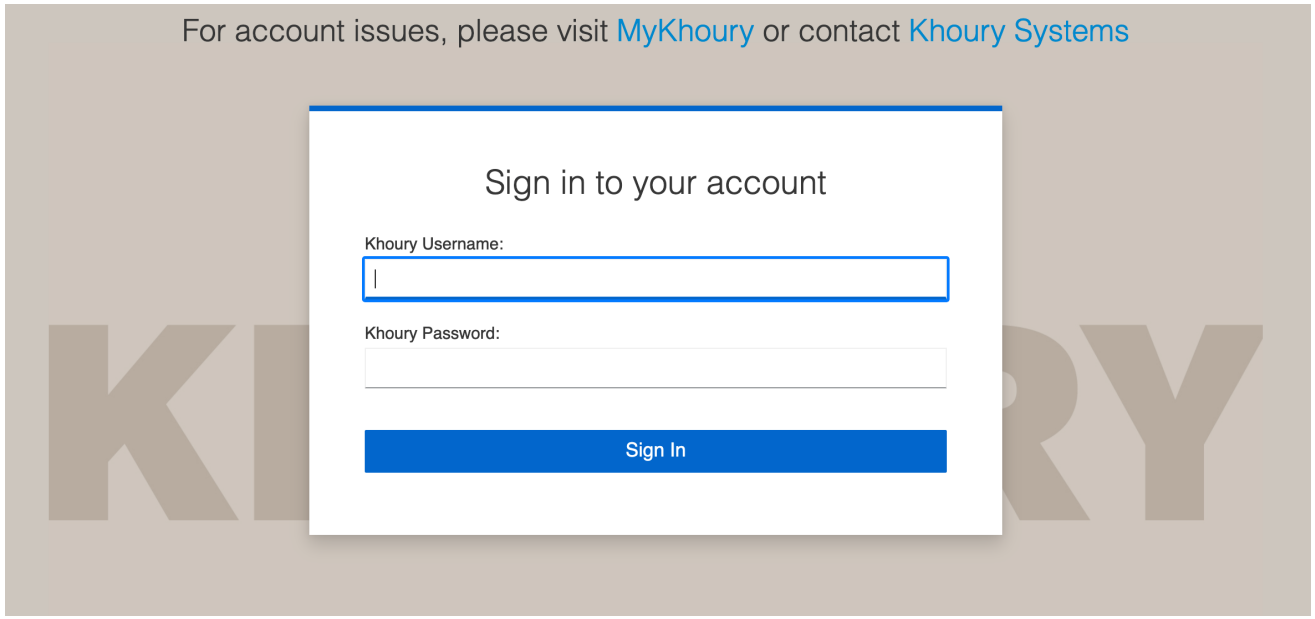
Installing Git on windows.

The Git Pro books has instructions: https://git-scm.com/book/en/v2/Getting-Started-Installing-Git



The Khoury Enterprise GIT Server

https://github.khoury.northeastern.edu/



Post-git Installation Commands

Your Identity

The first thing you should do when you install Git is to set your user name and email address. This is important because every Git commit uses this information, and it's immutably baked into the commits you start creating:

```
$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com
```

Again, you need to do this only once if you pass the `--global` option, because then Git will always use that information for anything you do on that system. If you want to override this with a different name or email address for specific projects, you can run the command without the `--global` option when you're in that project.

Many of the GUI tools will help you do this when you first run them.

Post-git Installation Commands

Your Editor

Now that your identity is set up, you can configure the default text editor that will be used when Git needs you to type in a message. If not configured, Git uses your system's default editor.

If you want to use a different text editor, such as Emacs, you can do the following:

```
$ git config --global core.editor emacs
```

On a Windows system, if you want to use a different text editor, you must specify the full path to its executable file. This can be different depending on how your editor is packaged.

In the case of Notepad++, a popular programming editor, you are likely to want to use the 32-bit version, since at the time of writing the 64-bit version doesn't support all plug-ins. If you are on a 32-bit Windows system, or you have a 64-bit editor on a 64-bit system, you'll type something like this:

```
$ git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -notabbar -nosession -noPlugin"
```

GIT Repository List

Why is my face so big?

rachlin
rachlin

Associate Teaching Professor Khoury
College of Computer Sciences
Northeastern University

Edit profile

Overview | Repositories | Projects | Stars

Find a repository... | Type | Language | Sort | New

ds3500_fa23 Public
Official Class Repo for DS3500 (Fall 2023)
13 Updated 6 hours ago
Star

ds3500_sp23 Private
Course Repository for DS3500 (Sp23)
Jupyter Notebook Updated on Apr 7
Star

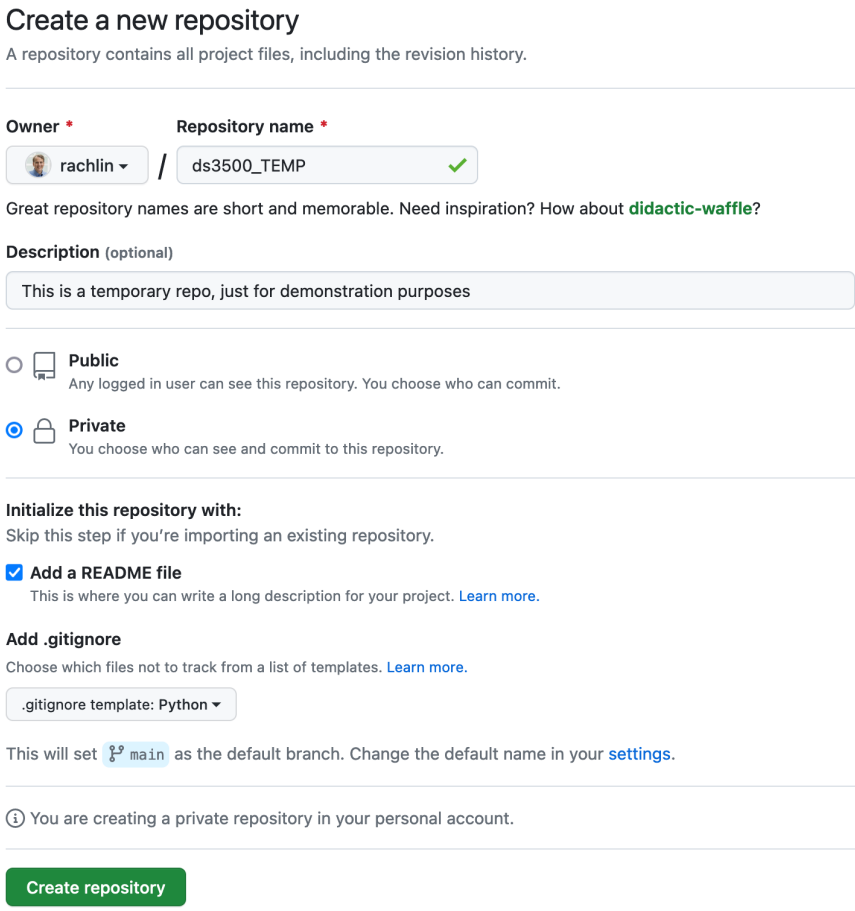
web Private
Khoury Website
HTML Updated on Feb 7
Star

Creating a new repository

Here I'm creating a PRIVATE repo called ds3500_TEMP.

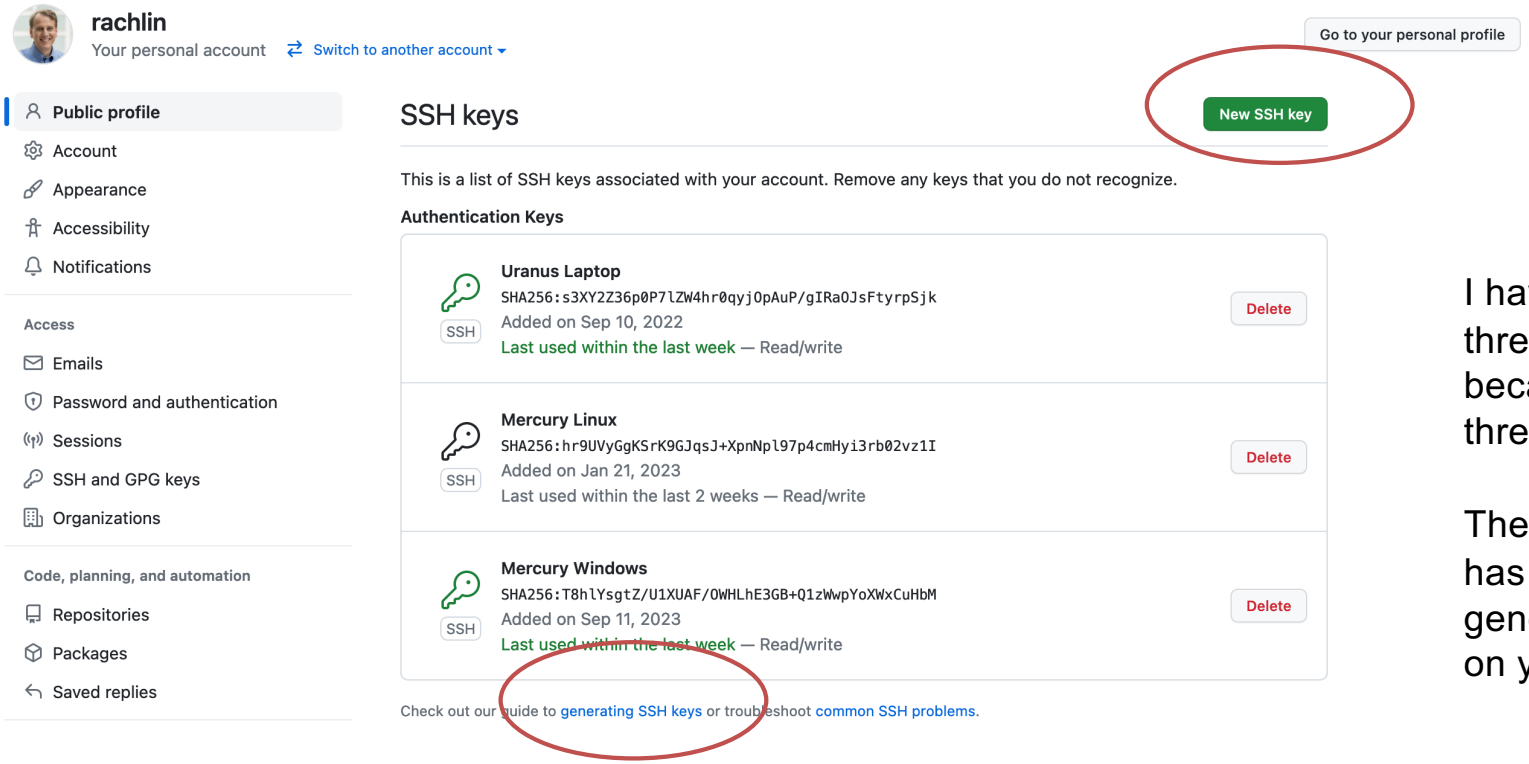
I'll have a README file.

Setting the .gitignore template to Python means that when we check for new files to add to the repo, git will ignore certain files and folders related to working with python projects or with particular IDEs but that are not part of the code base.



SSH Keys

Before I can clone my repo or run other git commands, I have to register my laptop with the Khoury Enterprise Git Server. **Go to Settings…. SSH and GPG Keys.**



I have registered three SSH keys because I use three machines.

The link at the bottom has instructions for generating SSH keys on your machine.

Generating SSH Keys

https://docs.github.com/en/enterprise-server@3.9/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

Generating a new SSH key

You can generate a new SSH key on your local machine. After you generate the key, you can add the public key to your account on your GitHub Enterprise Server instance to enable authentication for Git operations over SSH.

If you are a site administrator for your GitHub Enterprise Server instance, you can use the same key to grant yourself administrative SSH access to the instance. For more information, see "Accessing the administrative shell (SSH)."

1. Open Terminal.
2. Paste the text below, substituting in your GitHub Enterprise Server email address.

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Use your northeastern.edu email address.

Generating SSH Keys

https://docs.github.com/en/enterprise-server@3.9/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent

3. When you're prompted to "Enter a file in which to save the key", you can press **Enter** to accept the default file location. Please note that if you created SSH keys previously, ssh-keygen may ask you to rewrite another key, in which case we recommend creating a custom-named SSH key. To do so, type the default file location and replace id_ssh_keyname with your custom key name.

```
> Enter a file in which to save the key (/Users/YOU/.ssh/id_ALGORITHM): [Press enter]
```

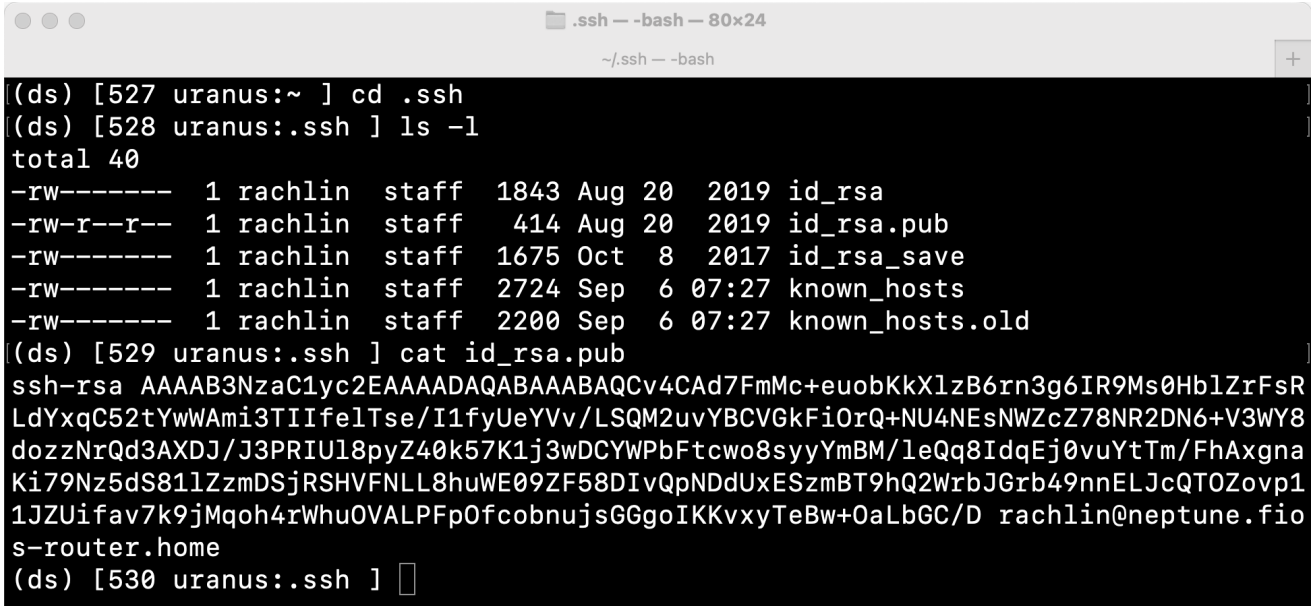
4. At the prompt, type a secure passphrase. For more information, see "Working with SSH key passphrases."

```
> Enter passphrase (empty for no passphrase): [Type a passphrase]
> Enter same passphrase again: [Type passphrase again]
```

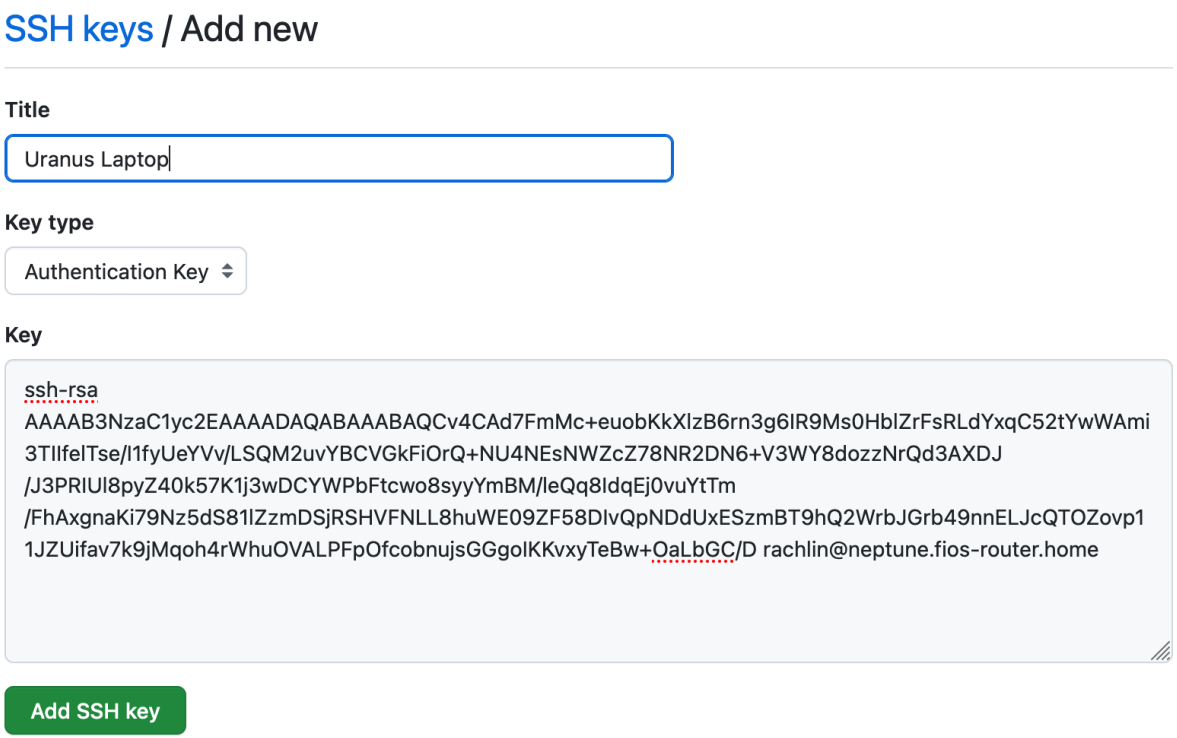
You can leave the passphrase empty for no passphrase.

Your public and private ssh key

Your newly-generated public and private SSH key live in your home directory inside a .ssh folder. (The dot before the folder name indicates that this is a hidden folder.). I don't mind showing you my public key (**id_rsa.pub**) but my private key (**id_rsa**) is for my eyes only! Your keys might be named something different. Just look for the two files, one of which has the **.pub** extension. THAT's your public key!



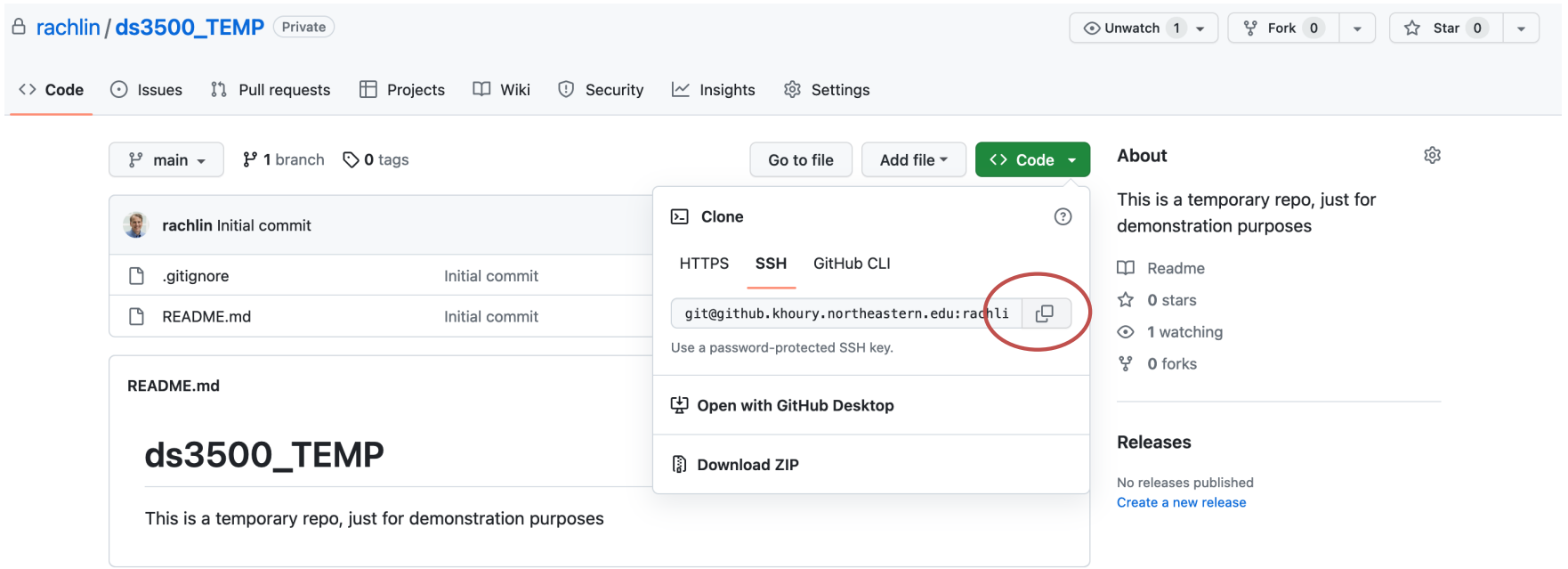
Registering the public SSH key



One the key is registered
I can run git commands without having to enter a username and password.

Let’s clone our test repo

Pull down the green “Code” button, select SSH, and copy that address to the clipboard.



Now clone the repo

```
(ds) [542 uranus:~ ] cd Documents
(ds) [543 uranus:Documents ] cd courses
(ds) [544 uranus:courses ] git clone git@github.khoury.northeastern.edu:rachlin/ds3500_TEMP.git
Cloning into 'ds3500_TEMP'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
Receiving objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
```

Pasted the repo address
after the “git clone” command.

Adding a file to the repo

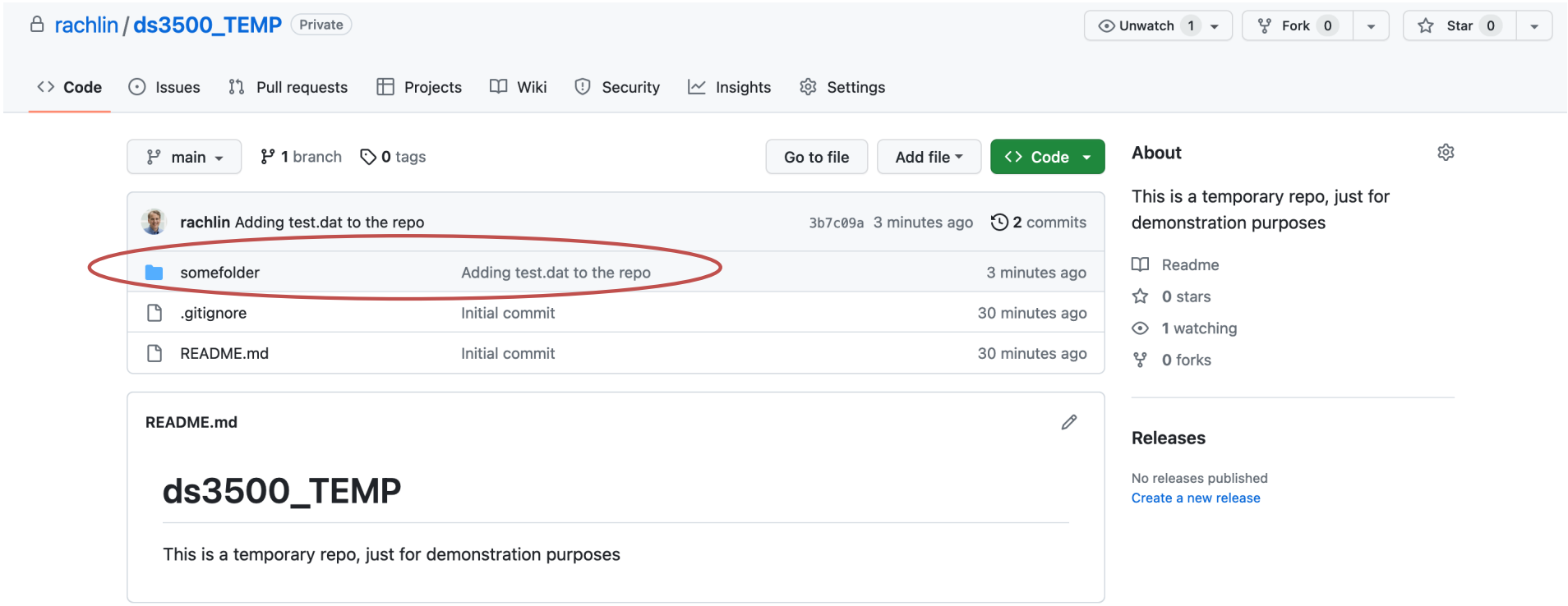
We create a subfolder and and a file called **test.dat** to the folder.

Next, we **add, commit,** and **push** the file in order to update the remote copy of the repo.

The Pro GIT book explains these commands in more detail. Please have a look.

```
(ds) [549 uranus:courses ] cd ds3500_TEMP/
(ds) [550 uranus:ds3500_TEMP ] ls
README.md
(ds) [551 uranus:ds3500_TEMP ] mkdir somefolder
(ds) [552 uranus:ds3500_TEMP ] cd somefolder
(ds) [553 uranus:somefolder ] echo "hello" > test.dat
(ds) [554 uranus:somefolder ] cat test.dat
hello
(ds) [555 uranus:somefolder ] git add test.dat
(ds) [556 uranus:somefolder ] git commit -m "Adding test.dat to the repo"
[main 3b7c09a] Adding test.dat to the repo
 1 file changed, 1 insertion(+)
 create mode 100644 somefolder/test.dat
(ds) [557 uranus:somefolder ] git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 382 bytes | 382.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.khoury.northeastern.edu:rachlin/ds3500_TEMP.git
   aeddc68..3b7c09a  main -> main
(ds) [558 uranus:somefolder ]
```

Refresh our repo page on the git server



Obligatory XKCD